

# Chapter 13

## Expressions (updated 23-Mar-2022)

This chapter describes the expression services in HCA. Expressions are used like in a traditional programming language to change the value of a variable.

This chapter covers these topics:

- Introduction to HCA expressions
- The Visual Programmer Compute and Compute Test elements
- The expression builder
- Managing variables
- Important uses of variables besides the Visual Programmer
- Error handling
- Expression syntax and built-in functions

In many cases the simpler variable values – Yes and No – and the three Visual Programmer elements – Make variable yes, Make variable No, and Not variable- are sufficient for applications. The Compute and Compute test elements are used for more sophisticated programming.

---

### Introduction to expressions

As described in the chapter on the Visual Programmer, HCA variables are usually used with simple Yes and No values. But in addition to those you can create and manipulate variables that can store text, numeric, Boolean, or date-time values.

Each variable can contain data of any type. HCA converts the data to the type it needs for the operator being evaluated. For example, consider these expressions:

```
a = 10
b = 20
c = "The result is" + (a + b)
d = #01-01-2001#
e = a - "8"
```

After these expressions are evaluated:

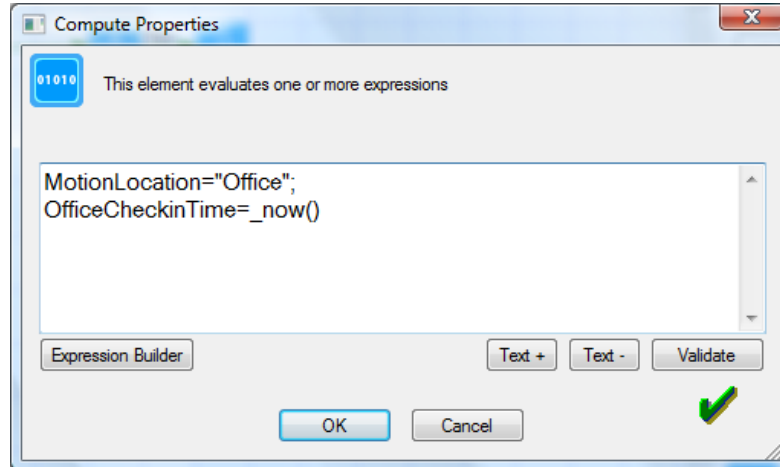
```
a is a number with value 10
b is a number with value 20
c is a string with value "The result is 30"
d is a date
e is a number with value 2
```

If you understand, or can learn about, how expressions in traditional programming languages like Visual Basic work, you will understand HCA expressions.

## Compute and Compute-Test visual programmer elements

To use these expressions two visual programmer elements are available: Compute and Compute Test.

The properties of the Compute element are:



In the Compute element is placed a series of expressions each separated by semicolons.

<variable name> = <expression> ;

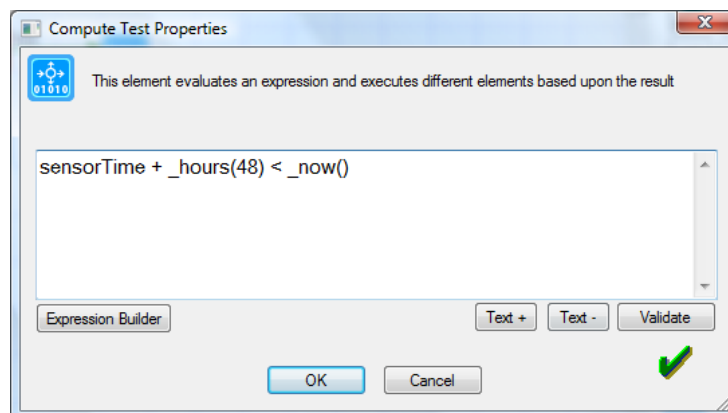
<variable name> = <expression> ;

...

<variable name> = <expression>

When the compute element is executed, the expressions are evaluated and the computed values assigned to the named variables. Expressions are executed in sequential order.

The Compute Test element contains a single expression that is evaluated to determine a yes or no value. If the value is "yes" the path marked "yes" in the program is taken from the Compute Test element, and likewise for "no".



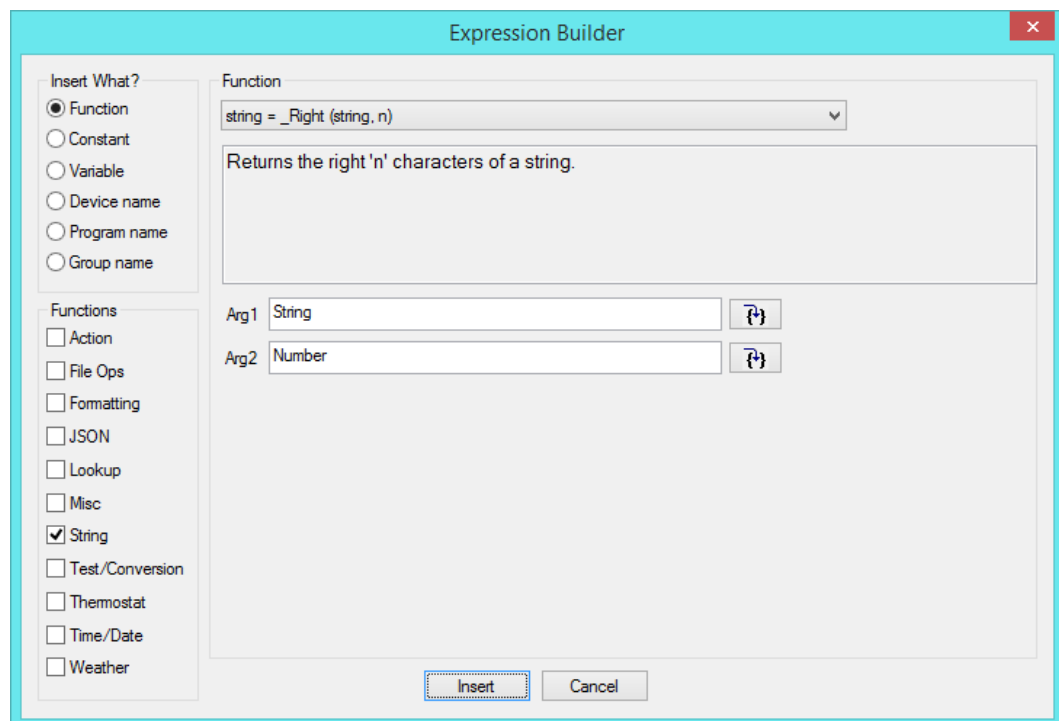
In both these elements the Validate button is used to check that the expression you have entered is correct – it matches the syntax that HCA expects.

A lot of work went into the Visual Programmer to allow HCA users to create programs without all the baggage of existing programming languages – careful syntax, programming terms and concepts. These two elements take a step back from that and leave you in the realm of the programmer. If you have never used, for example, Visual Basic, or all this sounds Greek to you, stick with simple yes and no variables managed with the visual programmer elements for them. You can do many wonderful things with them alone.

## Expression builder

To help you create expressions, rather than always having to refer to this documentation, HCA contains a tool called the Expression Builder.

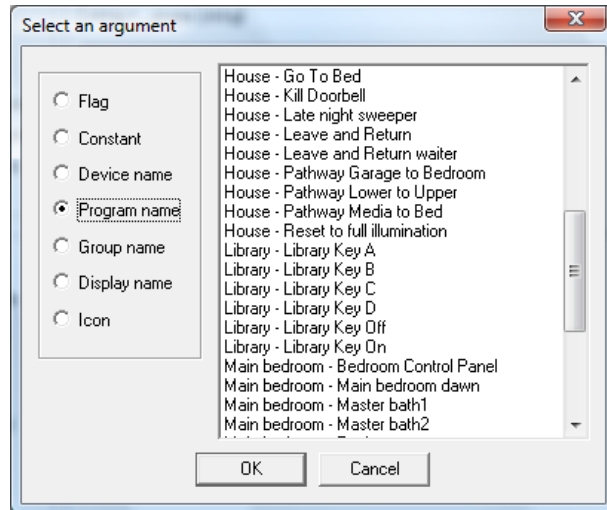
On dialogs where you enter an expression, a button labeled *Expression Builder* helps you compose your expression. Pressing this button opens the builder tool.



The “Insert what?” box specifies what sort of item you are inserting. The most common case is one of the HCA built-in functions.

The “Functions” box lets you limit the number of choices of the possible functions you have to choose from.

The third section of the dialog changes depending upon what you are inserting. In the picture above, a function is being inserted. Choose the name of the function in the dropdown and two things display: a short explanation of what it does, and the parameters to the function. In this example, the `_right` function takes two arguments. You can simply type in the two arguments, or to get more assistance, press the button next to the argument. This opens this dialog:



This dialog lets you insert common things that you may want to work with. Things like the names of the objects in your design, variables, and constants.

When you close the Expression Builder, the constructed expression is inserted into the text of the element properties at the cursor. Or it replaces the current selection if there is one.

---

## Managing Variables

An important point about variables is that they usually get created when expressions are evaluated. When a program is executed any new variables that are used in its expressions appear in the variables list in the variable inventory dialog.

The variable inventory dialog is described in the chapter on design tools.

---

## Other uses for expressions

In addition to using expressions in the Compute and Compute Test visual programmer elements, you can place expressions in other elements. Just enclose the expression in %'s. When the element is executed, the expression is evaluated and the result in text form replaces the % enclosed section. For example, to show the value of an expression, use this text in the ShowMessage element:

The value of beta is %beta1 + beta2%

If your string wants to display a percent sign, use two in the string:

Inside humidity is %humInside%% %

In these elements, an Embed Expression button appears. This lets you build an expression then encloses it in %'s when it places the expression into the element's properties.

To see in which elements an embedded expression can be used, see technical note titled "Parameter and expression use in program elements".

## Error Handling

Because these elements happen at a more complex layer of HCA than most elements, errors can happen that could not be detected in the Visual Programmer. If errors occur while executing these elements, the errors are logged, the Compute or Compute Test element is abandoned, and execution continues with the next element. In the case of the Compute Test element execution follows the Yes path. These errors show up with a red "P" marker and can be filtered as an Error.

Some of the possible errors are:

- Naming a device, thermostat or a magic module as an argument to a function and no device, thermostat, or magic module with that name in your design.
- Divide by zero.
- Using a weather function but no weather provider available.
- Trying to construct a date-time with something out of range. Like a month of 13.

## Expression Syntax

HCA expressions are very similar to expressions in any programming language like Visual Basic or VBScript. The usual operators are available:

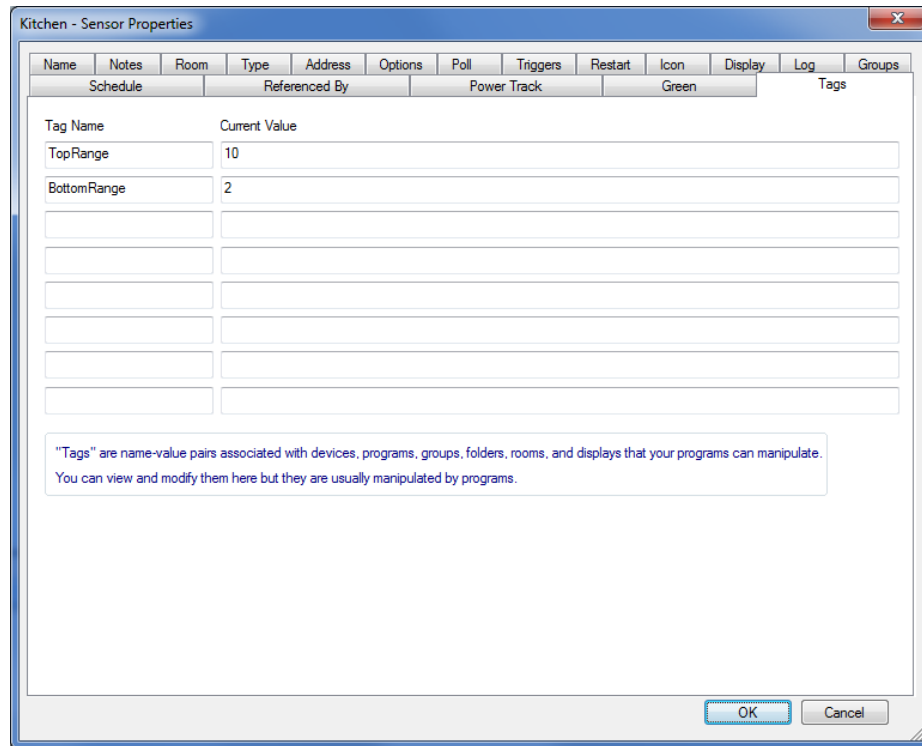
|                         |  |
|-------------------------|--|
| Comparison operators    | < > <= >= <> ==<br><b>Note</b> that the operator that checks for equality is 2 equal signs not one                       |
| Arithmetic operators    | + - * / mod - (unary minus) + (unary plus)   |
| Logical operators       | and or not eqv imp xor   |
| Binary operators        | Binary or operator is a Vertical bar  <br>Binary and operator is an Ampersand &<br>Binary not operator is a circumflex ^ |
| Date and Time constants | Enclosed in #'s as: #1/15/2001 07:19 AM#   |
| Boolean constants       | Yes No True False  |
| String constants        | Can be enclosed in single or double quotes   |
| Variables               | If the name of that variable has a blank in it, enclose the variable name in square brackets. For example [My Variable]  |

## Tags

A “tag” is simply a name-value pair associated with an object in your design. The name can be any valid HCA name and the value is any text string. You can associate up to 8 “tags” with a room, device, program, group, or keypad.

What are tags good for? That is up to your design. You can use them to associate any piece of data with an object. The key idea behind tags is that programs can manipulate them – adding, deleting, and seeing if a device has a tag or a specific value of a tag. In this way you can have programs save data in a tag of an object and then later another program can read it out. In some ways they are like global variables except that the value is specific to each object.

In the properties dialog there is a “Tags” tab where the current tags assigned to an object are shown with their values. Using this you can view and modify tags and their values from the UI.



In addition to manually viewing and changing, these expression functions are available and described in the Lookup Functions section below:

- ObjectTagGet
- ObjectTagSet
- ObjectTagDelete
- ObjectTagExists

---

## Expression functions

In the expressions used in the Compute and Compute-Test elements, you can use the built-in functions that HCA provides. Some of these are very general and can be found in almost any programming language, and others are specific to HCA.

Some notes on the function-by-function list below:

- In the list of functions below, some have optional parameters. These are designated by showing the parameter in []'s
- All functions return some value and in the Compute element you must assign that value to a variable. In the function list below if the result isn't useful, it is designated as "void".
- In creating expressions there are several functions that HCA provides. Some of these are very general and can be found in almost any programming language, and others are specific to HCA.
- All functions provided by HCA begin with the underscore character. If none of your variable names begin with an underscore, if in subsequent versions of HCA new functions are added, none of your variable names will conflict with any new function names. The case of the name is not important, so the left trim function can be written *Ltrim* or *ltrim*.

---

## String functions

The string functions are identical to the Visual Basic functions of the same name.

|                    |  |
|--------------------|--|
| <b>Name</b>        | Asc  |
| <b>Result type</b> | Number   |
| <b>Parameters</b>  | (string)   |
| <b>Action</b>      | Returns the ASCII value of the 1 <sup>st</sup> character of the string |
| <b>Example</b>     | _Asc("A") → 65   |

|                    |  |
|--------------------|--|
| <b>Name</b>        | Chr  |
| <b>Result type</b> | String   |
| <b>Parameters</b>  | (number, [# of repeats])   |
| <b>Action</b>      | Returns a one-character string of the ASCII character number unless the second argument is supplied then that tells the number of repeated characters to return. |
| <b>Example</b>     | _Chr(65) → "A"<br>_Chr(65, 3) → "AAA"  |

|                    |   |
|--------------------|---|
| <b>Name</b>        | DecToHex  |
| <b>Result type</b> | String  |
| <b>Parameters</b>  | (number, # of hex digits)   |
| <b>Action</b>      | Converts a number to a string where the value is expressed in hexadecimal |
| <b>Example</b>     | _DecToHex(100,2) → "64"   |

|                    |   |
|--------------------|---|
| <b>Name</b>        | HexToDec  |
| <b>Result type</b> | Number  |
| <b>Parameters</b>  | (string)  |
| <b>Action</b>      | Converts a string containing hex characters into a number. Undefined if there are non-hex characters in the string. |
| <b>Example</b>     | _HexToDec("64") → 100   |

|                    |  |
|--------------------|--|
| <b>Name</b>        | InStr  |
| <b>Result type</b> | Number   |
| <b>Parameters</b>  | (string, string)   |
| <b>Action</b>      | Finds the location of a string within another string. Returns the character position where the 1 <sup>st</sup> character is 1. Returns 0 if not found. |
| <b>Example</b>     | _InStr("webber", "bb") → 3   |

|                    |   |
|--------------------|---|
| <b>Name</b>        | LTrim   |
| <b>Result type</b> | String  |
| <b>Parameters</b>  | (string)  |
| <b>Action</b>      | Trims white-space characters from the left side of a string |
| <b>Example</b>     | <code>_LTrim(" example")</code> → "example"                 |

|                    |  |
|--------------------|--|
| <b>Name</b>        | Lcase                                    |
| <b>Result type</b> | String                                   |
| <b>Parameters</b>  | (string)                                 |
| <b>Action</b>      | Make a string lowercase                  |
| <b>Example</b>     | <code>_Lcase("Webber")</code> → "webber" |

|                    |   |
|--------------------|---|
| <b>Name</b>        | Left                                    |
| <b>Result type</b> | String                                  |
| <b>Parameters</b>  | (string, number)                        |
| <b>Action</b>      | Returns the leftmost 'n' characters     |
| <b>Example</b>     | <code>_Left("webber", 3)</code> → "web" |

|                    |  |
|--------------------|--|
| <b>Name</b>        | Len  |
| <b>Result type</b> | Number   |
| <b>Parameters</b>  | (string)                                       |
| <b>Action</b>      | Returns the number of characters in the string |
| <b>Example</b>     | <code>_Len("webber")</code> → 6                |

|                    |  |
|--------------------|--|
| <b>Name</b>        | Match  |
| <b>Result type</b> | Bool   |
| <b>Parameters</b>  | (string, pattern string)   |
| <b>Action</b>      | Performs a regular expression match between the supplied string and pattern. If the expression matches the pattern, then the function returns yes. |
| <b>Example</b>     | <code>_Match("00773", "00.*")</code> → yes   |



|                    |  |
|--------------------|--|
| <b>Name</b>        | Mid  |
| <b>Result type</b> | String   |
| <b>Parameters</b>  | (string, start at #, [# of characters])  |
| <b>Action</b>      | Returns characters from the start position for the supplied length. The first character in the string is 1. If the second argument is omitted the remainder of the start starting from the start position is returned. |
| <b>Example</b>     | <code>_Mid("webber", 2, 3) → "ebb"</code>  |

|                    |   |
|--------------------|---|
| <b>Name</b>        | RTrim   |
| <b>Result type</b> | String  |
| <b>Parameters</b>  | (string)  |
| <b>Action</b>      | Trims white-space characters from the right end of the string |
| <b>Example</b>     | <code>_RTrim("webber ") → "webber"</code>                     |

|                    |  |
|--------------------|--|
| <b>Name</b>        | Right  |
| <b>Result type</b> | String   |
| <b>Parameters</b>  | (string, # characters)                             |
| <b>Action</b>      | Returns the rightmost 'n' characters from a string |
| <b>Example</b>     | <code>_Right("webber", 2) → "er"</code>            |

|                    |   |
|--------------------|---|
| <b>Name</b>        | TextPiece   |
| <b>Result type</b> | String / Bool   |
| <b>Parameters</b>  | (string, piece #, delimiter string)   |
| <b>Action</b>      | Returns the nth piece of the string that contains sections of text between delimiters. If there is no nth piece in the string a bool of No is returned. |
| <b>Example</b>     | <code>_TextPiece("apple,banana,grape", 2, ",") → "banana"</code>  |

|                    |  |
|--------------------|--|
| <b>Name</b>        | TextReplace  |
| <b>Result type</b> | String   |
| <b>Parameters</b>  | (string, find string, replacement string)  |
| <b>Action</b>      | Replaces one string for another within a string.   |
| <b>Example</b>     | <code>_TextReplace("speed 10 mph", "mph", "miles per hour") → "speed 10 miles per hour"</code> |

|                    |  |
|--------------------|--|
| <b>Name</b>        | Trim   |
| <b>Result type</b> | String   |
| <b>Parameters</b>  | (string)   |
| <b>Action</b>      | Trims white-space characters from the left and right ends of a string. |
| <b>Example</b>     | <code>_Trim(" webber ") → "webber"</code>                              |

|                    |  |
|--------------------|--|
| <b>Name</b>        | Ucase                                    |
| <b>Result type</b> | String                                   |
| <b>Parameters</b>  | (string)                                 |
| <b>Action</b>      | Uppercases a string.                     |
| <b>Example</b>     | <code>_Ucase("webber") → "WEBBER"</code> |

---

## Test / Conversion functions

These functions are generally useful in creating programs, and many are like the corresponding Visual Basic functions.

|                    |  |
|--------------------|--|
| <b>Name</b>        | Abs                                      |
| <b>Result type</b> | Number                                   |
| <b>Parameters</b>  | (number)                                 |
| <b>Action</b>      | Returns the absolute value of the number |
| <b>Example</b>     | <code>_Abs(-87) → 87</code>              |

|                    |  |
|--------------------|--|
| <b>Name</b>        | Choose   |
| <b>Result type</b> | Any  |
| <b>Parameters</b>  | (number, any, any, ...)  |
| <b>Action</b>      | Returns as its result the Nth argument. The 1 <sup>st</sup> argument chooses which argument to be returned. The <i>any</i> arguments can be of any types and there can be up to 10 of them |
| <b>Example</b>     | <code>_Choose (3, "Jan", "Feb", "Mar", "Apr", "May") → "Mar"</code>  |

|                    |  |
|--------------------|--|
| <b>Name</b>        | IIF  |
| <b>Result type</b> | Any  |
| <b>Parameters</b>  | (bool, any1, any2)   |
| <b>Action</b>      | Returns any1 if the 1 <sup>st</sup> argument is yes. Otherwise returns any2. |
| <b>Example</b>     | <code>_IIF (yes, "red", "blue") → "red"</code>                               |

|                    |   |
|--------------------|---|
| <b>Name</b>        | Int   |
| <b>Result type</b> | Number  |
| <b>Parameters</b>  | (any)   |
| <b>Action</b>      | Converts the argument to a number if not already a number and returns it discarding any fractional part |
| <b>Example</b>     | <pre> _Int(412.87) → 412 _int("412.87") → 412 _int("abc") → 0 _int("") → 0 </pre>                       |

|                    |   |
|--------------------|---|
| <b>Name</b>        | IsBool  |
| <b>Result type</b> | Bool  |
| <b>Parameters</b>  | (any)   |
| <b>Action</b>      | Returns YES if the argument contains a bool value with <i>no</i> type coercion or an expression that evaluates to a bool. |
| <b>Example</b>     | <pre> _IsBool(TRUE) → Yes _IsBool(1) → No </pre>  |

|                    |   |
|--------------------|---|
| <b>Name</b>        | IsDate  |
| <b>Result type</b> | Bool  |
| <b>Parameters</b>  | (any)   |
| <b>Action</b>      | Returns YES if the argument contains a datetime value value with <i>no</i> type coercion or an expression that evaluates to a dateTime. |
| <b>Example</b>     | <pre> _IsDate("web") → No </pre>  |

|                    |   |
|--------------------|---|
| <b>Name</b>        | IsEven  |
| <b>Result type</b> | Bool  |
| <b>Parameters</b>  | (number)                                      |
| <b>Action</b>      | Returns YES is the argument is an even number |
| <b>Example</b>     | <pre> _IsEven(13) → No </pre>                 |

|                    |   |
|--------------------|---|
| <b>Name</b>        | IsNumber  |
| <b>Result type</b> | Bool  |
| <b>Parameters</b>  | (any)   |
| <b>Action</b>      | Returns YES if the argument contains a number value with <i>no</i> type coercion or an expression that evaluates to a number. |
| <b>Example</b>     | <pre> _IsNumber("412") → No _IsNumber(412) → Yes </pre>   |

|                    |  |
|--------------------|--|
| <b>Name</b>        | IsOdd  |
| <b>Result type</b> | Bool   |
| <b>Parameters</b>  | (number)                                     |
| <b>Action</b>      | Returns YES is the argument is an odd number |
| <b>Example</b>     | <code>_IsOdd(13)</code> → Yes                |

|                    |   |
|--------------------|---|
| <b>Name</b>        | IsText  |
| <b>Result type</b> | Bool  |
| <b>Parameters</b>  | (any)   |
| <b>Action</b>      | Returns YES if the argument contains a text value with <i>no</i> type coercion or an expression that evaluates to text. |
| <b>Example</b>     | <code>_IsText("hello there")</code> → Yes<br><code>_IsText(400)</code> → No   |

|                    |   |
|--------------------|---|
| <b>Name</b>        | Max   |
| <b>Result type</b> | Number  |
| <b>Parameters</b>  | (number, [number], [number], ...)                                     |
| <b>Action</b>      | Returns the maximum value of the arguments given. Up to 10 arguments. |
| <b>Example</b>     | <code>_Max(10, 50, 13, 17)</code> → 50                                |

|                    |   |
|--------------------|---|
| <b>Name</b>        | Min   |
| <b>Result type</b> | Number  |
| <b>Parameters</b>  | (number, [number], [number], ...)                                     |
| <b>Action</b>      | Returns the minimum value of the arguments given. Up to 10 arguments. |
| <b>Example</b>     | <code>_Min(10, 50, 13, 17)</code> → 10                                |

|                    |   |
|--------------------|---|
| <b>Name</b>        | Num   |
| <b>Result type</b> | Number  |
| <b>Parameters</b>  | (any)   |
| <b>Action</b>      | Converts the argument to a number if not already a number. Generally not needed because HCA converts between strings and numbers as needed. |
| <b>Example</b>     | <code>_Num("100")</code> → 100<br><code>_Num("abc")</code> → No<br><code>_Num("")</code> → No   |

|                    |  |
|--------------------|--|
| <b>Name</b>        | Round                                    |
| <b>Result type</b> | Number                                   |
| <b>Parameters</b>  | (number)                                 |
| <b>Action</b>      | Rounds the number to the nearest integer |
| <b>Example</b>     | <code>_Round(402.6)</code> → 403         |

---

Time and date functions

For the examples below, assume that the current time is 02:12:45 pm and the current date is Friday 28-September-2018

|                    |  |
|--------------------|--|
| <b>Name</b>        | Date   |
| <b>Result type</b> | DateTime   |
| <b>Parameters</b>  | (year #, month #, day#)  |
| <b>Action</b>      | Constructs a datetime value from the supplied year, month, and day. Year is the 4-digit year, month is 1-12, day is 1-31 |
| <b>Example</b>     | <code>_Date(2018, 9, 28)</code> → 28-Sep-2018  |

|                    |  |
|--------------------|--|
| <b>Name</b>        | DateTime   |
| <b>Result type</b> | DateTime   |
| <b>Parameters</b>  | (year #, month #, day#, hour#, minute#, second#)   |
| <b>Action</b>      | Constructs a datetime value from the supplied year, month, day, hour, minute, second. Year is the 4-digit year, month is 1-12, day is 1-31, hour is 0-23, minute is 0-59, and second is 0-59 |
| <b>Example</b>     | <code>_DateTime(2018, 9, 28, 14, 16, 30)</code> → 28-Sep-2018 2:16:30pm  |

|                    |                                 |
|--------------------|---------------------------------|
| <b>Name</b>        | Day                             |
| <b>Result type</b> | Number                          |
| <b>Parameters</b>  | (dateTime)                      |
| <b>Action</b>      | Returns the day from a dateTime |
| <b>Example</b>     | <code>_Day(_Now())</code> → 28  |

|                    |  |
|--------------------|--|
| <b>Name</b>        | DayOfWeek  |
| <b>Result type</b> | Number   |
| <b>Parameters</b>  | (dateTime)   |
| <b>Action</b>      | Returns the weekday from a dateTime as a number from 1 to 7, where 1 is Sunday |
| <b>Example</b>     | <code>_DayOfWeek(_Now())</code> → 6  |

|                    |   |
|--------------------|---|
| <b>Name</b>        | DayOfYear   |
| <b>Result type</b> | Number  |
| <b>Parameters</b>  | (dateTime)  |
| <b>Action</b>      | Returns the ordinal number of the day of the year |
| <b>Example</b>     | <code>_DayOfYear(_now())</code> → 271             |

|                    |   |
|--------------------|---|
| <b>Name</b>        | Days  |
| <b>Result type</b> | DateTimeSpan  |
| <b>Parameters</b>  | (#days)   |
| <b>Action</b>      | Creates a date time span equal to the number of days                    |
| <b>Example</b>     | <code>_Now() + _Days(1)</code> → Saturday 28-September-2018 02:12:45 PM |

|                    |                                 |
|--------------------|---------------------------------|
| <b>Name</b>        | Hour                            |
| <b>Result type</b> | Number                          |
| <b>Parameters</b>  | (dateTime)                      |
| <b>Action</b>      | Returns the hour of a date time |
| <b>Example</b>     | <code>_Hour(_Now())</code> → 14 |

|                    |  |
|--------------------|--|
| <b>Name</b>        | Hours  |
| <b>Result type</b> | DateTimeSpan   |
| <b>Parameters</b>  | (#hours)   |
| <b>Action</b>      | Creates a date time span equal to the number of hours                  |
| <b>Example</b>     | <code>_Now() + _Hours(2)</code> → Friday 28-September-2018 04:12:45 PM |

|                    |  |
|--------------------|--|
| <b>Name</b>        | LocalToUTC   |
| <b>Result type</b> | DateTime   |
| <b>Parameters</b>  | (dateTime)   |
| <b>Action</b>      | Convert a local time to UTC. The conversion is done by Windows so the local time zone and DST setting as set in Windows is used. |
| <b>Example</b>     | <code>_LocalToUTC(_Now())</code> → Friday 28-September-2020 04:12:45 PM  |

|                    |                                   |
|--------------------|-----------------------------------|
| <b>Name</b>        | Minute                            |
| <b>Result type</b> | Number                            |
| <b>Parameters</b>  | (dateTime)                        |
| <b>Action</b>      | Returns the minute of a date time |
| <b>Example</b>     | <code>_Minute(_Now())</code> → 12 |

|                    |   |
|--------------------|---|
| <b>Name</b>        | Minutes   |
| <b>Result type</b> | DateTimeSpan  |
| <b>Parameters</b>  | (#minutes)  |
| <b>Action</b>      | Creates a dateTimeSpan equal to the number of minutes                     |
| <b>Example</b>     | <code>_Now() + _Minutes(10)</code> → Friday 28-September-2018 02:22:45 PM |

|                    |                                  |
|--------------------|----------------------------------|
| <b>Name</b>        | Month                            |
| <b>Result type</b> | Number                           |
| <b>Parameters</b>  | (dateTime)                       |
| <b>Action</b>      | Returns the month of a date time |
| <b>Example</b>     | <code>_Month(_Now())</code> → 9  |

|                    |  |
|--------------------|--|
| <b>Name</b>        | MonthName  |
| <b>Result type</b> | String   |
| <b>Parameters</b>  | (month#, [use full name?])   |
| <b>Action</b>      | Returns the name of the month as a text string. If the optional 2 <sup>nd</sup> argument is supplied and is yes, the full name is generated otherwise an abbreviation is used. |
| <b>Example</b>     | <code>_MonthName(_now(), yes)</code> → “September”   |

|                    |  |
|--------------------|--|
| <b>Name</b>        | Now  |
| <b>Result type</b> | DateTime   |
| <b>Parameters</b>  | none   |
| <b>Action</b>      | Returns as a datetime the current date and time            |
| <b>Example</b>     | <code>_Now()</code> → Friday 28-September-2018 02:12:45 PM |

|                    |   |
|--------------------|---|
| <b>Name</b>        | ParseTime   |
| <b>Result type</b> | DateTime / Bool   |
| <b>Parameters</b>  | (“datetime text”)   |
| <b>Action</b>      | Parses the argument into a dateTime. Returns that datetime if the parse worked. Returns a Bool No if not. |
| <b>Example</b>     | <code>_ParseTime(“10/4/2018 5:34:44 AM”)</code> → #10/4/2018 5:34:44 AM#                                  |

|                    |  |
|--------------------|--|
| <b>Name</b>        | Second                                 |
| <b>Result type</b> | Number                                 |
| <b>Parameters</b>  | (dateTime)                             |
| <b>Action</b>      | Returns the second part of a date time |
| <b>Example</b>     | <code>_Second(_Now())</code> → 45      |

|                    |   |
|--------------------|---|
| <b>Name</b>        | Seconds   |
| <b>Result type</b> | DateTimeSpan  |
| <b>Parameters</b>  | (#seconds)  |
| <b>Action</b>      | Creates a dateTimeSpan equal to the number of seconds                     |
| <b>Example</b>     | <code>_Now() + _Seconds(10)</code> → Friday 28-September-2018 02:12:55 PM |



|                    |   |
|--------------------|---|
| <b>Name</b>        | Sunrise   |
| <b>Result type</b> | DateTime  |
| <b>Parameters</b>  | None  |
| <b>Action</b>      | Returns the sunrise time for the current location |
| <b>Example</b>     | <code>_Sunrise()</code> → 9/28/2018 7:03 AM       |

|                    |  |
|--------------------|--|
| <b>Name</b>        | Sunset   |
| <b>Result type</b> | DateTime   |
| <b>Parameters</b>  | None   |
| <b>Action</b>      | Returns the sunset time for the current location |
| <b>Example</b>     | <code>_Sunset()</code> → 9/28/2018 6:57 PM       |

|                    |  |
|--------------------|--|
| <b>Name</b>        | Time   |
| <b>Result type</b> | DateTime   |
| <b>Parameters</b>  | (hour, minute, second)                                     |
| <b>Action</b>      | Creates a dateTime with the given hour, minute, and second |
| <b>Example</b>     | <code>_Time(14, 20, 0)</code> → 02:20:00 PM                |

|                    |   |
|--------------------|---|
| <b>Name</b>        | TimeSpan  |
| <b>Result type</b> | DateTimeSpan  |
| <b>Parameters</b>  | (days, hours, minutes, seconds)   |
| <b>Action</b>      | Creates a dateTimeSpan with the given values  |
| <b>Example</b>     | <code>_Now() + _TimeSpan(0, 1, 20, 0)</code> → Friday 28-September-2018 03:32:45 PM |

|                    |  |
|--------------------|--|
| <b>Name</b>        | TotalHours   |
| <b>Result type</b> | Number   |
| <b>Parameters</b>  | (dateTimeSpan)   |
| <b>Action</b>      | Returns the total number of hours represented by the time span |
| <b>Example</b>     | <code>_TotalHours(_TimeSpan(0, 1, 30, 0))</code> → 1.5         |

|                    |  |
|--------------------|--|
| <b>Name</b>        | TotalMinutes   |
| <b>Result type</b> | Number   |
| <b>Parameters</b>  | (dateTimeSpan)   |
| <b>Action</b>      | Returns the total number of hours represented by the time span |
| <b>Example</b>     | <code>_TotalMinutes(_TimeSpan(0, 1, 30, 0))</code> → 90        |

|                    |  |
|--------------------|--|
| <b>Name</b>        | TotalSeconds   |
| <b>Result type</b> | Number   |
| <b>Parameters</b>  | (dateTimeSpan)   |
| <b>Action</b>      | Returns the total number of seconds represented by the time span |
| <b>Example</b>     | <code>_TotalSeconds(_TimeSpan(0, 1, 30, 0))</code> → 5400        |

|                    |   |
|--------------------|---|
| <b>Name</b>        | UTCToLocal  |
| <b>Result type</b> | DateTime  |
| <b>Parameters</b>  | (dateTime)  |
| <b>Action</b>      | Convert a UTC time to local time. The conversion is done by Windows so the local time zone and DST setting as set in Windows is used. |
| <b>Example</b>     | <code>_UTCToLocal(_Now())</code> → Friday 28-September-2020 04:12:45 PM   |

|                    |  |
|--------------------|--|
| <b>Name</b>        | Weekday  |
| <b>Result type</b> | String   |
| <b>Parameters</b>  | (#days ago)  |
| <b>Action</b>      | Returns the three-letter abbreviation of the weekday. (0) = today, (1) = yesterday, (2) = 2 days ago, etc. |
| <b>Example</b>     | <code>_Weekday(2)</code> → “Wed”   |

|                    |  |
|--------------------|--|
| <b>Name</b>        | WeekdayName  |
| <b>Result type</b> | String   |
| <b>Parameters</b>  | (number, [full name?])   |
| <b>Action</b>      | Returns a string of the weekday name. Sunday is numbered 1, and Saturday is numbered 7. If the second argument is supplied and is YES, the full name is returned, otherwise the 3-letter abbreviation is returned. |
| <b>Example</b>     | <code>_WeekdayName(_DayOfWeek(_Now()), No)</code> → “Fri”  |

|                    |                                   |
|--------------------|-----------------------------------|
| <b>Name</b>        | Year                              |
| <b>Result type</b> | Number                            |
| <b>Parameters</b>  | (dateTime)                        |
| <b>Action</b>      | Returns the year of the dateTime  |
| <b>Example</b>     | <code>_Year(_Now())</code> → 2018 |

**There are four major uses of the time functions in the Compute element. These are:**

- Determine how long something took. This is done by:

```
t = _Now()
```

... do something...

```
timeItTook = _Now() - t
```

- Add or subtract from the current time to generate a date-time in the past or future:

```
TwentyFourHoursAgo = _Now() - _days(1)
```

```
SixAndAHalfHoursAgo = _Now() - _timeSpan(0, 6, 30, 0)
```

- Compose a date-time from its component parts:

```
t = _DateTime(2018, 9, 28, 14, 12, 45)
```

- Format a date-time to a string:

```
s = _FormatTime(_Now(), "$d-$b-$y $H:$M")
```

This would show as "28-Sep-18 14:12"

---

## Formatting functions

|                    |   |
|--------------------|---|
| <b>Name</b>        | FormatInt   |
| <b>Result type</b> | String  |
| <b>Parameters</b>  | (#, #digits, [leading zeros?])  |
| <b>Action</b>      | Convert a number to a string with no fractional part. If the 3rd argument is supplied as YES, the string is formatted with leading zeros. |
| <b>Example</b>     | <code>_FormatInt(100.5, 4, 1) → "0100"</code>   |

|                    |   |
|--------------------|---|
| <b>Name</b>        | FormatNum   |
| <b>Result type</b> | String  |
| <b>Parameters</b>  | (#, #decimal places)  |
| <b>Action</b>      | Converts the number to a string with the given number of digits after the decimal point |
| <b>Example</b>     | <code>_FormatNum(1.6764, 1) → "1.6"</code>  |

|                    |   |
|--------------------|---|
| <b>Name</b>        | FormatPatten  |
| <b>Result type</b> | String  |
| <b>Parameters</b>  | (#, "pattern")  |
| <b>Action</b>      | Convert a number to a string according to the pattern. The pattern uses the same characters as that used for the C programming language sprintf function. Look online for references to sprintf. The only difference is the \$ character is used instead of the % character in the pattern. |
| <b>Example</b>     | <code>_FormatPattern (412.543, "\$f.1") → "412.5"</code>  |

|                    |   |
|--------------------|---|
| <b>Name</b>        | FormatTime  |
| <b>Result type</b> | String  |
| <b>Parameters</b>  | (dateTime)  |
| <b>Action</b>      | Returns a string of the date-time formatted according to the pattern. The pattern is a string made up of replacements from the following table. |
| <b>Example</b>     | <code>_FormatTime(_now(), "\$d-\$b \$l:\$M \$p")</code> → "28-Sep 2:12 pm"  |

FormatTime pattern characters:

| Pattern marker | Meaning  |
|----------------|--|
| \$a            | Abbreviated weekday name   |
| \$A            | Full weekday name  |
| \$b            | Abbreviated month name   |
| \$B            | Full month name  |
| \$c            | Date and time appropriate for locale                                       |
| \$d            | Day of month as number (01-31)   |
| \$H            | Hour in 24-hour format (00-23)   |
| \$I            | Hour in 12-hour format (01-12)   |
| \$j            | Day of year as a number (001-366)  |
| \$m            | Month as a number (01-12)  |
| \$M            | Minutes as a number (00-59)  |
| \$p            | Current locale's AM/PM indicator for 12-hour clock                         |
| \$S            | Second as a number (00-59)   |
| \$U            | Week of year as a number, with Sunday as the first day of the week (00-51) |
| \$w            | Weekday as a number (0-6). Sunday is 0.                                    |
| \$W            | Week of year as number with Monday as the first day of the week (00-51)    |
| \$x            | Date representation appropriate for locale                                 |
| \$X            | Time representation appropriate for locale                                 |
| \$y            | Year without century as a number (00-99)                                   |
| \$Y            | Year with century as number  |
| \$z or \$Z     | Time-zone name or abbreviation. Blank if not known                         |
| \$             | Dollar sign  |

---

## Action functions

These functions perform actions on your devices, programs, and groups in your design.

|                    |   |
|--------------------|---|
| <b>Name</b>        | AutoOffTime   |
| <b>Result type</b> | Date-time / Bool  |
| <b>Parameters</b>  | ("device name")   |
| <b>Action</b>      | Returns the date-time when the named device/room auto off timer will expire.<br>If there is no auto off timer running for the named device/room, a Boolean value of NO is returned. This can be tested for with <code>_IsBool</code> function.<br><br>Note: This function only reports the auto off time. You can make changes to the auto off timer and settings for a device using the Auto-Off programmer element. |
| <b>Example</b>     | <code>_AutoOffTime("Kitchen – Lights")</code> → No  |

|                    |  |
|--------------------|--|
| <b>Name</b>        | ChangeSchedule   |
| <b>Result type</b> | Bool   |
| <b>Parameters</b>  | ("schedule entry name", code#, time1, [time2])   |
| <b>Action</b>      | Modifies the schedule entry with the given name. The codes are:<br>0 = Change On Time, 1 = Change Off Time, 2 = Change On and Off Time.<br><br>With code 2 you must supply the 4 <sup>th</sup> argument.<br><br>Returns YES if the modification is made, NO otherwise. |
| <b>Example</b>     | <code>_ChangeSchedule("OutsideSet", 0, _Time(20,0,0))</code> → Yes<br>This sets the on time of the schedule entry to 8pm   |

|                    |   |
|--------------------|---|
| <b>Name</b>        | ModifySchedule  |
| <b>Result type</b> | Bool  |
| <b>Parameters</b>  | See below   |
| <b>Action</b>      | <code>_ModifySchedule</code> is a more capable replacement for <code>_ChangeSchedule</code><br><br>The first argument to <code>ModifySchedule</code> is a code that defines what the operation is. This is followed by up to 7 additional arguments<br><br>Arg1: Code 1 = Create schedule<br>Arg2: Schedule name<br>Arg3: [Optional] Name of parent schedule<br><br>Arg1: Code 2 = Delete schedule<br>Arg2: Schedule name<br><br>Arg1: Code 3 = Clear schedule – remove all schedule entries<br>Arg2: Schedule name |

|                |   |
|----------------|---|
|                | <p>Arg1: Code 4 = Create schedule entry<br/>         Arg2: Schedule name<br/>         Arg3: Entry name (can be "" for an unnamed entry)<br/>         Arg4: Target name (device, program, or group being scheduled)<br/>         Arg5: On code (0=nop, 1=atTime, 2=sunrise, 3=sunset)<br/>         Arg6: On data (if Arg5 is 1 then data is a time, if Arg5 is 2 or 3 then data is a signed # of minutes)<br/>         Arg7: Off code (0=nop, 1=atTime, 2=sunrise, 3=sunset)<br/>         Arg8: Off data (if Arg7 is 1 then data is a time, if Arg7 is 2 or 3 then data is a signed # of minutes)</p> <p>An ON entry is created if arg5 and arg6 are supplied and arg7 and arg8 are omitted<br/>         An OFF entry is created if arg5 and arg6 are omitted or arg5 is 0, and arg7 and arg8 are supplied<br/>         An ON-OFF entry is created if arg5, arg6, arg7, and arg8 are supplied</p> <p>Arg1: Code 5 = Delete schedule entry<br/>         Arg2: entry name</p> <p>Arg1: Code 6 = Modify schedule entry<br/>         Arg2: Entry name<br/>         Arg3: On code (0=nop, 1=atTime, 2=sunrise, 3=sunset)<br/>         Arg4: On data (if Arg3 is 1 then data is a time, if Arg3 is 2 or 3 then data is a signed # of minutes)<br/>         Arg5: Off code (0=nop, 1=atTime, 2=sunrise, 3=sunset)<br/>         Arg6: Off data (if Arg5 is 1 then data is a time, if Arg5 is 2 or 3 then data is a signed # of minutes)</p> <p>When modifying an existing schedule entry, the ON or OFF clause is modified if data is supplied or that part of the schedule entry is unmodified if the ON/OFF codes are 0 (for no operation)</p> <p>Returns YES if the action was possible, No otherwise. For example, a deletion of a schedule isn't allowed if it is in use in a change schedule VP element.</p> |
| <b>Example</b> | <code>_ModifySchedule(4, "My Schedule", "", "Kitchen-Lights", 1, _Time(22,30,0))</code> → Yes   |

|                    |  |
|--------------------|--|
| <b>Name</b>        | CurrentScene   |
| <b>Result type</b> | String   |
| <b>Parameters</b>  | ("device name")  |
| <b>Action</b>      | Returns the current scene, if known, for the device.         |
| <b>Example</b>     | <code>_CurrentScene("kitchen - lights")</code> → "Nighttime" |

|                    |   |
|--------------------|---|
| <b>Name</b>        | CurrentSchedule                           |
| <b>Result type</b> | String                                    |
| <b>Parameters</b>  | none                                      |
| <b>Action</b>      | Returns the name of the current schedule. |
| <b>Example</b>     | <code>_CurrentSchedule()</code> → “Away”  |

|                    |   |
|--------------------|---|
| <b>Name</b>        | DayNight  |
| <b>Result type</b> | Bool  |
| <b>Parameters</b>  | (“Device name”, code#, data)  |
| <b>Action</b>      | Reprograms device if possible to change configuration<br>Code 1: Change switch on-level. Data is percent 0-100<br>Code 2: Change LED level. Data is “High”, “Medium”, “Low”, “Faint”, or “None”<br>Code 3: Change backlighting. Data is “Off” or “On”<br><br>Note: Not all devices can be reprogrammed. |
| <b>Example</b>     | <code>_DayNight(“Kitchen – Lights”, 1, 20)</code> → Yes   |

|                    |  |
|--------------------|--|
| <b>Name</b>        | DimDownPercent   |
| <b>Result type</b> | Number   |
| <b>Parameters</b>  | (“device name”, percent)   |
| <b>Action</b>      | Decreases the named object percentage by the supplied amount. Returns 0 if able to send the command, -1 if not. Note that a return code of 0 only means the command was sent and not that the device received and acted upon it. |
| <b>Example</b>     | <code>_DimDownPercent(“Kitchen – Lights”, 20)</code> → 0   |

|                    |  |
|--------------------|--|
| <b>Name</b>        | DimPercent   |
| <b>Result type</b> | Number   |
| <b>Parameters</b>  | (“device name”, [request status?])   |
| <b>Action</b>      | Returns the dim percentage of the named object.<br>If the 2 <sup>nd</sup> parameter is NO or omitted, the evaluation is based upon internal HCA state. If the 2 <sup>nd</sup> parameter is YES and if the device is 2-way, its state is requested to determine the percent returned. |
| <b>Example</b>     | <code>_DimPercent(“Kitchen – Lights”)</code> → 80  |

|                    |   |
|--------------------|---|
| <b>Name</b>        | DimToPercent  |
| <b>Result type</b> | Number  |
| <b>Parameters</b>  | ("device name", percent)  |
| <b>Action</b>      | Controls the named object the supplied percentage. Returns 0 if able to send the command, -1 if not. Note that a return code of 0 only means the command was sent and not that the device received and acted upon it. |
| <b>Example</b>     | _DimToPercent("Kitchen – Lights", 50) → 0   |

|                    |  |
|--------------------|--|
| <b>Name</b>        | DimUpPercent   |
| <b>Result type</b> | Number   |
| <b>Parameters</b>  | ("device name", percent)   |
| <b>Action</b>      | Increases the named object percentage by the supplied amount. Returns 0 if able to send the command, -1 if not. Note that a return code of 0 only means the command was sent and not that the device received and acted upon it. |
| <b>Example</b>     | _DimUpPercent("Kitchen – lights", 10) → 0  |

|                    |   |
|--------------------|---|
| <b>Name</b>        | Hue   |
| <b>Result type</b> | Bool – Yes if the operation worked, No if not   |
| <b>Parameters</b>  | See below   |
| <b>Action</b>      | <p>Function version of the Hue element. The first argument to the function is a code that determines the operation.</p> <p>Arg1: Code 0 = On<br/>Arg2: Device or Hue group name</p> <p>Arg1: Code 1 = Off<br/>Arg2: Device or Hue group name</p> <p>Arg1: Code 2 = Set to percent<br/>Arg2: Device or Hue group name<br/>Arg3: Percent</p> <p>Arg1: Code 3 = Set to color<br/>Arg2: Device or Hue group name<br/>Arg3: Color name or if providing the color by HSB: H<br/>Arg4: S<br/>Arg5: B</p> <p>Arg1: Code 4 = Set group to scene<br/>Arg2: Hue group name<br/>Arg3: Hue scene name<br/>Arg4: Percent, if omitted defaults to 100%</p> |
| <b>Example</b>     | _Hue(2, "Kitchen – lights", 50) → Yes   |



|                    |  |
|--------------------|--|
| <b>Name</b>        | HueColorCreate   |
| <b>Result type</b> | void   |
| <b>Parameters</b>  | ("color name", hue, sat, brt, [update if already exists])  |
| <b>Action</b>      | Adds or updates a color in the colors available for the Hue function and the Hue programmer element. If the final optional argument is present and if YES then the color is updated. Otherwise, if it exists it is not updated.<br><br>This is useful for library programs to make colors available. |
| <b>Example</b>     | <code>_HueColorCreate("Pink", 56018, 98, 216)</code>   |

|                    |   |
|--------------------|---|
| <b>Name</b>        | IconChange  |
| <b>Result type</b> | Void  |
| <b>Parameters</b>  | ("name", ["icon name"], ["display name"])   |
| <b>Action</b>      | Change the displayed icon for a device, program, group, or display.<br><br>The 2nd parameter is the name of the icon to change to. If omitted, the original icon selected for the object is restored.<br><br>The 3rd parameter is the name of the effected display. If omitted, then all displays with an icon for this object are changed. |
| <b>Example</b>     | <code>_IconChange("Kitchen – lights", "Appliance")</code>   |

|                    |  |
|--------------------|--|
| <b>Name</b>        | IconChangeEx   |
| <b>Result type</b> | Void   |
| <b>Parameters</b>  | ("name", code, [arg3], [arg4], [arg5])   |
| <b>Action</b>      | Change the displayed icon for a device, program, group, or display. The 1 <sup>st</sup> argument is the object whose icon is modified. The 2 <sup>nd</sup> argument is a code.<br><br>code=0 - Returns the icon to HCA control and resets the icon, icon representation, label, and annotation text<br><br>code=1 - Changes the icon<br>arg3: Icon name<br>arg4: <Optional>Icon representation (0=On, 1=Off, 2=Dim)<br><br>If the option 4 <sup>th</sup> argument is present that sets the representation of the icon as specified. The state of the object doesn't change the icon representation.<br><br>If the optional 4 <sup>th</sup> argument is omitted, then while the icon image is changed, the state of the object controls the representation of the icon.<br><br>code=2 - Changes the label text below the icon<br>arg3: Label text<br><br>code=3 - Change the text to the right of the icon if a theme with that feature is in use<br>arg3: Text<br><br>code=4 - Displays text over the icon ("annotation")<br>arg3: Text (limited to 48 characters)<br>arg4: <optional>Text size in points. If omitted uses 18<br>arg5: <optional>RGB of the text color. If omitted uses black (Hint: Use <code>_RGB</code> function) |

|                |   |
|----------------|---|
|                | To make multiple lines, embed a newline character in the text where it breaks like this:<br>“abc” + <code>_chr(10)</code> + “def” |
| <b>Example</b> | <code>_IconChangeEx(“Kitchen – Lights”, 2, “Ceiling Lights”)</code>   |

|                    |  |
|--------------------|--|
| <b>Name</b>        | IconExists   |
| <b>Result type</b> | Bool   |
| <b>Parameters</b>  | (“icon name”, [“theme name”])  |
| <b>Action</b>      | Returns YES if the named icon exists in the named theme. If the theme argument is omitted, then the default theme is used. |
| <b>Example</b>     | <code>_IconExists(“Appliance”) → YES</code>  |

|                    |   |
|--------------------|---|
| <b>Name</b>        | InsteonBeep   |
| <b>Result type</b> | Void  |
| <b>Parameters</b>  | (“device name”, code)   |
| <b>Action</b>      | Sends a command to the named Insteon device that should cause it to beep. |
| <b>Example</b>     | <code>_InsteonBeep(“Kitchen – Lights”, 0)</code>                          |

|                    |  |
|--------------------|--|
| <b>Name</b>        | IsCurrentSchedule  |
| <b>Result type</b> | Bool   |
| <b>Parameters</b>  | (“schedule name”)  |
| <b>Action</b>      | Returns YES if the named schedule is the current schedule. |
| <b>Example</b>     | <code>_IsCurrentSchedule(“Away”) → Yes</code>              |

|                    |  |
|--------------------|--|
| <b>Name</b>        | IsDim  |
| <b>Result type</b> | Bool / Number  |
| <b>Parameters</b>  | (“Device name”, [request status?])   |
| <b>Action</b>      | Returns YES if the named object is at 1% to 99%<br>If the 2nd parameter is NO or omitted, the evaluation is based upon internal HCA state.<br>If the 2nd parameter is YES, and the object supports status, its state is requested to determine the return value. If the device doesn’t respond to the status poll, a -1 is returned. |
| <b>Example</b>     | <code>_IsDim(“Kitchen – lights”) → Yes</code>  |

|                    |   |
|--------------------|---|
| <b>Name</b>        | IsDisabled  |
| <b>Result type</b> | Bool  |
| <b>Parameters</b>  | (“name”)  |
| <b>Action</b>      | Returns Yes if the object is disabled.            |
| <b>Example</b>     | <code>_IsDisabled(“Kitchen – Lights”) → No</code> |

|                    |   |
|--------------------|---|
| <b>Name</b>        | IsInErrorState  |
| <b>Result type</b> | Bool  |
| <b>Parameters</b>  | ("name")  |
| <b>Action</b>      | Returns yes if the object is in an error state due to a previous communication failure with the device. |
| <b>Example</b>     | _IsInErrorState("Kitchen – Lights") → No  |

|                    |   |
|--------------------|---|
| <b>Name</b>        | IsOff   |
| <b>Result type</b> | Bool / Number   |
| <b>Parameters</b>  | ("Device name", [request status?])  |
| <b>Action</b>      | Returns YES if the named object is OFF, NO otherwise.<br>If the 2nd parameter is NO or omitted, the evaluation is based upon internal HCA state.<br>If the 2nd parameter is YES and the object supports status, its state is requested to determine the return value. If the device doesn't respond to the status poll, a -1 is returned. |
| <b>Example</b>     | _IsOff("Kitchen Lights") → No   |

|                    |  |
|--------------------|--|
| <b>Name</b>        | IsOn   |
| <b>Result type</b> | Bool / Number  |
| <b>Parameters</b>  | ("Device name", [request status?])   |
| <b>Action</b>      | Returns YES if the named object is ON, NO otherwise.<br>If the 2nd parameter is NO or omitted, the evaluation is based upon internal HCA state.<br>If the 2nd parameter is YES and the object supports status, its state is requested to determine the return value. If the device doesn't respond to the status poll, a -1 is returned. |
| <b>Example</b>     | _IsOn("Kitchen Lights") → Yes  |

|                    |   |
|--------------------|---|
| <b>Name</b>        | IsRunning                                       |
| <b>Result type</b> | Bool  |
| <b>Parameters</b>  | ("Program name")                                |
| <b>Action</b>      | Returns YES if the program is currently running |
| <b>Example</b>     | _IsRunning("Home – Driveway Alert") → Yes       |

|                    |  |
|--------------------|--|
| <b>Name</b>        | IsSuspended                                  |
| <b>Result type</b> | Bool   |
| <b>Parameters</b>  | ("name")                                     |
| <b>Action</b>      | Returns YES if the named object is suspended |
| <b>Example</b>     | _IsSuspended("Kitchen – lights") → No        |

|                    |   |
|--------------------|---|
| <b>Name</b>        | Off   |
| <b>Result type</b> | Number  |
| <b>Parameters</b>  | ("name", [button#])   |
| <b>Action</b>      | Controls the named object to OFF. If the optional 2nd argument is supplied, it designates a keypad button indicator. Returns 0 if able to send the command, -1 if not. Note that a return code of 0 only means the command was sent and not that the device received and acted upon it. |
| <b>Example</b>     | _Off ("Kitchen – lights") → 0   |

|                    |  |
|--------------------|--|
| <b>Name</b>        | On   |
| <b>Result type</b> | Number   |
| <b>Parameters</b>  | ("name", [button#])  |
| <b>Action</b>      | Controls the named object to ON. If the optional 2nd argument is supplied, it designates a keypad button indicator. Returns 0 if able to send the command, -1 if not. Note that a return code of 0 only means the command was sent and not that the device received and acted upon it. |
| <b>Example</b>     | _On ("Kitchen – lights") → 0   |

|                    |  |
|--------------------|--|
| <b>Name</b>        | SetCurrentState  |
| <b>Result type</b> | Void   |
| <b>Parameters</b>  | ("name", percent#)   |
| <b>Action</b>      | Change the internal maintained state of the named object to the percent supplied. The device is not actually communicated with. Also updates the icons for the device.<br>0% = OFF, 100% = ON, 1%-99% = Dim. |
| <b>Example</b>     | _SetCurrentState("Kitchen – lights", 80)   |

|                    |  |
|--------------------|--|
| <b>Name</b>        | SetHomeMode  |
| <b>Result type</b> | Bool   |
| <b>Parameters</b>  | (code#)  |
| <b>Action</b>      | Set new home mode using code values:<br>0 = Home & Awake, 1 = Home & Asleep, 2 = Away, 3 = 4 <sup>th</sup> mode<br><br>Returns YES if successful, NO if not. |
| <b>Example</b>     | _SetHomeMode(1) → Yes  |

|                    |   |
|--------------------|---|
| <b>Name</b>        | SetRunAgainTime   |
| <b>Result type</b> | Void  |
| <b>Parameters</b>  | (date-time, ["program name"])   |
| <b>Action</b>      | <p>Requests a program to start at the specified time. Does, in effect, the same action as the Auto-Start configuration on the program's Advanced Options tab but using a computed time rather than a fixed time as specified there.</p> <p>If the time is given as 0, any previous request for start is cancelled.</p> <p>If the 2nd argument is provided, the named program is the target otherwise it applies to the running program.</p> |
| <b>Example</b>     | <code>_SetRunAgainTime(_now() + _hours(1))</code>   |

|                    |  |
|--------------------|--|
| <b>Name</b>        | SetToScene   |
| <b>Result type</b> | String   |
| <b>Parameters</b>  | ("name", "scene name")   |
| <b>Action</b>      | Controls a device to a named scene. Returns the current scene before the named scene is set. |
| <b>Example</b>     | <code>_SetToScene("Kitchen – lights", "nighttime") → ""</code>                               |

|                    |   |
|--------------------|---|
| <b>Name</b>        | StartProgram  |
| <b>Result type</b> | void  |
| <b>Parameters</b>  | ("program name", [arg1], [arg2], ... [arg8])  |
| <b>Action</b>      | <p>If the program doesn't "return a value" – option on program "Advanced Options" tab – then starts a program running independently from the program containing this function. Does not hold until that program is complete. If additional arguments are provided, they are passed to the started program.</p> <p>If the program being started returns a result – option enabled on the program "Advanced Options" tab -then the program that contains the Start-Program waits until the started program to complete and the result of the StartProgram function is the value generated by the started program.</p> |
| <b>Example</b>     | <code>_StartProgram("Outside – HandleLights", 100, 10)</code>   |

---

 Thermostat functions

|                    |   |
|--------------------|---|
| <b>Name</b>        | GetThermostat   |
| <b>Result type</b> | Bool / Number   |
| <b>Parameters</b>  | ("thermostat device name", code#)   |
| <b>Action</b>      | <p>Retrieves the thermostat setting given by the code from the table below. Returns a number if the operation worked and a bool of No if it didn't.</p> <p>It is up to program that uses this function to request only settings supported by the thermostat and for the setpoints only when in the correct mode.</p> <p>The return value is the setting retrieved or an error. Use the <code>_IsBool</code> on the result to determine if you have received the requested data or an error.</p> |
| <b>Example</b>     | <code>_GetThermostat("Home - Thermostat", 1) → 68</code>  |

|                    |   |
|--------------------|---|
| <b>Name</b>        | SetThermostat   |
| <b>Result type</b> | Bool / Number   |
| <b>Parameters</b>  | ("thermostat device name", code#, value#, [code#], [value#], ...)   |
| <b>Action</b>      | <p>Changes a thermostat setting given by the code to the value. Codes are given in the table below.</p> <p>Can change from 1 to 5 settings at once with optional code value pairs.</p> <p>Returns a Yes if the operation worked and No if it didn't. Use the <code>_IsBool</code> on the result to determine if the operation worked or an error.</p> <p>It is up to program that uses this function to change only settings supported by the thermostat and to change the setpoints only when in the correct mode.</p> |
| <b>Example</b>     | <code>_SetThermostat("Home - Thermostat", 2, 68) → Yes</code>   |

| Code | Setting               | Returned value  |
|------|-----------------------|---|
| 0    | Temperature           | Integer value   |
| 1    | Heat Setpoint         | Integer value   |
| 2    | Mode                  | Off = 0, Heat = 1, Cool = 2, Auto = 3   |
| 3    | Fan                   | 0 = On, 1 = Off   |
| 4    | Economy               | 0 = On, 1 = Off   |
| 5    | Aux Heat              | 0 = On, 1 = Off   |
| 6    | Humidity              | Integer value   |
| 7    | Cool Setpoint         | Integer value   |
| 8    | Has Leaf (NEST only)  | 0 = On, 1 = Off   |
| 13   | Nest Mode (NEST only) | 0 = Away, 1 = Home<br>When changing the NEST mode, it changes all thermostats in the structure associated with the thermostat being controlled. |

|                    |  |
|--------------------|--|
| <b>Name</b>        | TempDecode   |
| <b>Result type</b> | Bool / Number  |
| <b>Parameters</b>  | (unitcode#, level#)  |
| <b>Action</b>      | Returns the temperature using the RCS X10 decode table for temperature reporting. Useful for legacy thermostats. |
| <b>Example</b>     |  |

---

## Weather functions

|                    |  |
|--------------------|--|
| <b>Name</b>        | WeatherGet   |
| <b>Result type</b> | Number   |
| <b>Parameters</b>  | ("data item name", [code#], [#hours])  |
| <b>Action</b>      | <p>This function retrieves the named weather item in the units the provider is configured for.</p> <p>The item name, provided as a string, is the name of the weather item to retrieve. The item names are the same as used in the weather-test element: "Temperature", "Apparent Temperature", "Dew Point", etc.</p> <p>The code is optional and is specified as: 0=current, 1=max, 2=min, 3=average.</p> <p>The #hours argument is optional and is specified as a positive number for forecast data and as a negative number for historical data.</p> <p>If the data isn't available the value -999 is returned.</p> |
| <b>Example</b>     | <pre>_WeatherGet("Wind speed", 1, -3) → 10.9 Gets the max wind speed in the last 3 hours  _WeatherGet("Wind speed", 1, +3) → 3.2 Gets the max wind speed expected in the next 3 hours</pre>  |

|                    |   |
|--------------------|---|
| <b>Name</b>        | DarkSky   |
| <b>Result type</b> | String  |
| <b>Parameters</b>  | ("path1", ["path2"], ["path3"], ...)  |
| <b>Action</b>      | <p>Retrieves data from the dark sky weather provider using the supplied path. There can be from 1 to 10 arguments which are the path through the JSON. The path elements are not case sensitive.</p> <p>A program error happens if the weather provider is not dark sky. If the requested data isn't available -999 is returned. When retrieving a key whose value is time, a conversion is automatically done between dark sky time (unix time) and Windows time.</p> <p>See the Dark Sky technical note for more information.</p> |
| <b>Example</b>     | <pre>_DarkSky("currently", "pressure") → "1010.34"</pre>  |

|                    |  |
|--------------------|--|
| <b>Name</b>        | BarometerConvert   |
| <b>Result type</b> | Number   |
| <b>Parameters</b>  | (number, from units code#, to units code#)   |
| <b>Action</b>      | Converts between barometer units. The unit codes are:<br>Inches = 0, Millimeters = 1, Millbars = 2, HectoPascals = 3 |
| <b>Example</b>     | _BarometerConvert (32, 0, 2) → 1015  |

|                    |   |
|--------------------|---|
| <b>Name</b>        | BarometerUnits                          |
| <b>Result type</b> | String                                  |
| <b>Parameters</b>  | None                                    |
| <b>Action</b>      | Returns a string of the barometer units |
| <b>Example</b>     | _BarometerUnits() → "mb"                |

|                    |  |
|--------------------|--|
| <b>Name</b>        | HumidityUnits                          |
| <b>Result type</b> | String                                 |
| <b>Parameters</b>  | None                                   |
| <b>Action</b>      | Returns a string of the humidity units |
| <b>Example</b>     | _HumidityUnits() → "%"                 |

|                    |  |
|--------------------|--|
| <b>Name</b>        | RainConvert  |
| <b>Result type</b> | Number   |
| <b>Parameters</b>  | (Number, from units code#, to units code#)                                 |
| <b>Action</b>      | Converts between rain units. The codes are:<br>Inches = 0, Millimeters = 1 |
| <b>Example</b>     | _RainConvert (1, 0, 1) → 25.4  |

|                    |  |
|--------------------|--|
| <b>Name</b>        | RainUnits                                    |
| <b>Result type</b> | String                                       |
| <b>Parameters</b>  | None   |
| <b>Action</b>      | Returns a string of the rain intensity units |
| <b>Example</b>     | _RainUnits() → "in/hr"                       |

|                    |   |
|--------------------|---|
| <b>Name</b>        | SolarRadiationUnits                           |
| <b>Result type</b> | String  |
| <b>Parameters</b>  | None  |
| <b>Action</b>      | Returns a string of the solar radiation units |
| <b>Example</b>     | _SolarRadiationUnits() → "W/sq m"             |



|                    |  |
|--------------------|--|
| <b>Name</b>        | TempConvert  |
| <b>Result type</b> | Number   |
| <b>Parameters</b>  | (number, from units code#, to units code#)                         |
| <b>Action</b>      | Converts between temperature units. The codes are:<br>F = 0, C = 1 |
| <b>Example</b>     | _TempConvert(32, 0, 1) → 0   |

|                    |   |
|--------------------|---|
| <b>Name</b>        | TempUnits                                 |
| <b>Result type</b> | String                                    |
| <b>Parameters</b>  | None                                      |
| <b>Action</b>      | Returns a string of the temperature units |
| <b>Example</b>     | _TempUnits() → "F"                        |

|                    |                                  |
|--------------------|----------------------------------|
| <b>Name</b>        | UVUnits                          |
| <b>Result type</b> | String                           |
| <b>Parameters</b>  | None                             |
| <b>Action</b>      | Returns a string of the UV units |
| <b>Example</b>     | _UVUnits() → "UV Index"          |

|                    |  |
|--------------------|--|
| <b>Name</b>        | WindDirUnits                                 |
| <b>Result type</b> | String                                       |
| <b>Parameters</b>  | None   |
| <b>Action</b>      | Returns a string of the wind direction units |
| <b>Example</b>     | _WindDirUnits() → "degrees"                  |

|                    |  |
|--------------------|--|
| <b>Name</b>        | WindDirection  |
| <b>Result type</b> | String   |
| <b>Parameters</b>  | (number)   |
| <b>Action</b>      | Changes a wind direction in degrees into a string of the form: N, NNE, NE, ENE, etc. |
| <b>Example</b>     | _WindDirection (90) → "E"  |

|                    |  |
|--------------------|--|
| <b>Name</b>        | WindSpeedConvert   |
| <b>Result type</b> | Number   |
| <b>Parameters</b>  | (number, from units code#, to units code#)   |
| <b>Action</b>      | Converts between wind speed units. The codes are:<br>Miles per hour = 0, Knots = 1, Kilometers per hour = 2, Meters per second = 3 |
| <b>Example</b>     | _WindSpeedConvert (10, 0, 2) → 16.09   |

|                    |  |
|--------------------|--|
| <b>Name</b>        | WindSpeedUnits                           |
| <b>Result type</b> | String                                   |
| <b>Parameters</b>  | None                                     |
| <b>Action</b>      | Returns a string of the wind speed units |
| <b>Example</b>     | <code>_WindSpeedUnits()</code> → “m/s”   |

---

## File operation functions

This category of functions comprises a set of functions that operate on disk-based files. HCA allows a maximum of 16 files to be open at one time.

|                    |   |
|--------------------|---|
| <b>Name</b>        | FileOpen  |
| <b>Result type</b> | Number  |
| <b>Parameters</b>  | ("path", open option)   |
| <b>Action</b>      | <p>Opens a file so that the ReadString and WriteString functions can be used</p> <p>Option 0 = Open for Reading</p> <p>Option 1 = Open for Writing</p> <p>Option 2 = Open for writing and writes append to the end</p> <p>If the file can't be opened the result is -1 otherwise a number from 0 to 15 inclusive.</p> |
| <b>Example</b>     | hFile = _FileOpen("myFile.txt", 1)  |

|                    |  |
|--------------------|--|
| <b>Name</b>        | FileClose  |
| <b>Result type</b> | Void   |
| <b>Parameters</b>  | (file handle#)   |
| <b>Action</b>      | Closes a file previously opened with FileOpen. The argument is the value returned from FileOpen. |
| <b>Example</b>     | Void = _FileClose(hFile)   |

|                    |  |
|--------------------|--|
| <b>Name</b>        | FileExists   |
| <b>Result type</b> | Bool   |
| <b>Parameters</b>  | ("path to the file")   |
| <b>Action</b>      | Determines if a file exists and returns Yes if it does and No if it doesn't. |
| <b>Example</b>     | _FileExists("myFile.txt") → No   |

|                    |  |
|--------------------|--|
| <b>Name</b>        | FileLoad   |
| <b>Result type</b> | String / Bool  |
| <b>Parameters</b>  | ("path to the file", code#)  |
| <b>Action</b>      | <p>Opens the file at the supplied path, reads the contents and returns the file contents as one string. The options are:</p> <p>0 = copy CR-LF characters in the file into the result string, 1 = replace CR-LF characters in the file with a single blank in the result string</p> <p>Returns YES if worked, NO otherwise. Test with _IsText or _IsBool to check</p> <p>There is no limit on the size of the file, but a really big file will probably break HCA.</p> |
| <b>Example</b>     | _FileLoad("my file.txt", 0) → "The file contents"  |

|                    |   |
|--------------------|---|
| <b>Name</b>        | FileReadString  |
| <b>Result type</b> | String / Bool   |
| <b>Parameters</b>  | (file handle #)   |
| <b>Action</b>      | <p>Reads from a file opened by <code>_FileOpen</code>.</p> <p>The handle parameter is the number returned from <code>_FileOpen</code></p> <p>If there is data remaining in the file, the result is the string read from the file.</p> <p>If there is no data remaining in the file, then the result is a bool value of No.</p> <p>The result can be tested with the <code>IsText</code> or <code>IsBool</code> functions.</p> |
| <b>Example</b>     | <code>_FileReadString (hFile) → "A line from the file"</code>   |

|                    |   |
|--------------------|---|
| <b>Name</b>        | FileWriteString   |
| <b>Result type</b> | Number  |
| <b>Parameters</b>  | (file handle #, "data to write")  |
| <b>Action</b>      | <p>Writes to a file opened by <code>_FileOpen</code>. The handle parameter is the number returned from <code>_FileOpen</code>. The number of characters written to the file is returned. If the string ends with an ASCII newline character, then the line in the file ends with both a CR and NL characters. Create a newline with a <code>_chr(10)</code></p> |
| <b>Example</b>     | <code>_FileWriteString (hFile, "Hello web" + _chr(10))</code>   |

|                    |   |
|--------------------|---|
| <b>Name</b>        | FileDelete  |
| <b>Result type</b> | Bool  |
| <b>Parameters</b>  | (path)  |
| <b>Action</b>      | <p>Deletes a file. If the delete is successful – the file existed and is now removed – Yes is returned, No otherwise.</p> |
| <b>Example</b>     | <code>_FileDelete ("myfile.txt")</code>   |

|                    |  |
|--------------------|--|
| <b>Name</b>        | HCAFolder  |
| <b>Result type</b> | String   |
| <b>Parameters</b>  | None   |
| <b>Action</b>      | Returns the path to the HCA sub-folder in your documents area. |
| <b>Example</b>     | <code>_HCAFolder() → "c:\users\kimberly\documents\HCA"</code>  |

---

**JSON functions**

JSON is a method of encoding data. HCA has several functions that work with JSON. Refer to the JSON technical note for more information and examples.

|                    |   |
|--------------------|---|
| <b>Name</b>        | Json  |
| <b>Result type</b> | String / Bool   |
| <b>Parameters</b>  | (handle, "key1", ["key2"], ["key3"]...)   |
| <b>Action</b>      | Retrieves a value from the parsed JSON starting at the current position. There can be from 2 to 10 arguments.<br><br>arg1 is the handle returned by _JsonOpen<br><br>arg2 - arg10 are the key names to find at each level.<br><br>Returns the extracted data if found, NO otherwise. Use _IsBool to check the result. |
| <b>Example</b>     | _Json(handle, "1", "action", "xy", "0") → "0.4573"  |

|                    |   |
|--------------------|---|
| <b>Name</b>        | JsonOpen  |
| <b>Result type</b> | Number / Bool   |
| <b>Parameters</b>  | ("text")  |
| <b>Action</b>      | Parses the JSON text into an internal form and returns a handle to it. If the parse fails, then the result is NO. Use _IsBool to check for failure. A valid return is a number 1 to 16 inclusive. |
| <b>Example</b>     | Text = "{ \"Color\" : \"blue\" , \"State\" : true}";<br>hJson = _JsonOpen(text);  |

|                    |   |
|--------------------|---|
| <b>Name</b>        | JsonClose   |
| <b>Result type</b> | Void  |
| <b>Parameters</b>  | (# returned from json open)   |
| <b>Action</b>      | Releases the parsed form of the JSON. The handle is what was returned by _JsonOpen. |
| <b>Example</b>     | _JsonClose(hJson)   |

|                    |  |
|--------------------|--|
| <b>Name</b>        | JsonDown   |
| <b>Result type</b> | Bool   |
| <b>Parameters</b>  | (# returned from json open)  |
| <b>Action</b>      | Moves the current position to the first child of the current key in the parsed JSON. The handle is what was returned by _JsonOpen.<br>Returns YES if there if the move was successful, NO otherwise. |
| <b>Example</b>     | _JsonDown(hJson) → Yes   |

|                    |  |
|--------------------|--|
| <b>Name</b>        | JsonNext   |
| <b>Result type</b> | Bool   |
| <b>Parameters</b>  | (# returned from json open)  |
| <b>Action</b>      | Moves the current position to the next key in the parsed JSON. The handle is what was returned by <code>_JsonOpen</code> .<br>Returns YES if there if the move was successful, NO otherwise. |
| <b>Example</b>     | <code>_JsonNext(hJson) → Yes</code>  |

|                    |  |
|--------------------|--|
| <b>Name</b>        | JsonPrev   |
| <b>Result type</b> | Bool   |
| <b>Parameters</b>  | (# returned from json open)  |
| <b>Action</b>      | Moves the current position to the previous key in the parsed JSON. The handle is what was returned by <code>_JsonOpen</code> .<br>Returns YES if there if the move was successful, NO otherwise. |
| <b>Example</b>     | <code>_JsonPrev(hJson) → Yes</code>  |

|                    |   |
|--------------------|---|
| <b>Name</b>        | JsonUp  |
| <b>Result type</b> | Bool  |
| <b>Parameters</b>  | (# returned from json open)   |
| <b>Action</b>      | Moves the current position to the parent of the current key in the parsed JSON. The handle is what was returned by <code>_JsonOpen</code> .<br>Returns YES if there if the move was successful, NO otherwise. |
| <b>Example</b>     | <code>_JsonUp(hJson) → Yes</code>   |

|                    |   |
|--------------------|---|
| <b>Name</b>        | JsonCurrentKey  |
| <b>Result type</b> | text  |
| <b>Parameters</b>  | (# returned from json open)   |
| <b>Action</b>      | Returns the key from the current position in the JSON as moved by <code>JsonUp</code> , <code>JsonDown</code> , <code>JsonNext</code> , <code>JsonPrev</code> |
| <b>Example</b>     | <code>_JsonCurrentKey(hJson) → "Color"</code>   |

|                    |   |
|--------------------|---|
| <b>Name</b>        | JsonCurrentValue  |
| <b>Result type</b> | text  |
| <b>Parameters</b>  | (# returned from json open)   |
| <b>Action</b>      | Returns the value from the current position in the JSON as moved by <code>JsonUp</code> , <code>JsonDown</code> , <code>JsonNext</code> , <code>JsonPrev</code> |
| <b>Example</b>     | <code>_JsonCurrentValue(hJson) → "Blue"</code>  |

---

## Lookup functions

This category of functions is useful for working with the elements of your design.

|                    |   |
|--------------------|---|
| <b>Name</b>        | AddressForDevice  |
| <b>Result type</b> | String  |
| <b>Parameters</b>  | ("device name")   |
| <b>Action</b>      | Returns the "address" of the device. Formatted as per the protocol of the device. |
| <b>Example</b>     | <code>_AddressForDevice("Den – Plug")</code> → "192.168.0.182"                    |

|                    |  |
|--------------------|--|
| <b>Name</b>        | CurrentWattage   |
| <b>Result type</b> | Number   |
| <b>Parameters</b>  | (["device or room name"])  |
| <b>Action</b>      | Returns the current wattage used by the device or room. With no argument supplied it returns the whole home current wattage. |
| <b>Example</b>     | <code>_CurrentWattage("Kitchen – Lights")</code> → 400   |

|                    |  |
|--------------------|--|
| <b>Name</b>        | DesignOpen   |
| <b>Result type</b> | Number   |
| <b>Parameters</b>  | (code#, ["room-folder name"], ["tag name"], ["tag value"])   |
| <b>Action</b>      | <p>Works with <code>_DesignName</code> and <code>_DesignClose</code> to allow you to operate on each element in your design. Which elements depends upon the code used.</p> <p>Code=1 devices, code=2 programs, code=3 groups, code=4 rooms &amp; folders, code=5 variables, code=6 only rooms, code=7 only folders, code=8 only displays.</p> <p>If the optional 2nd argument is used for codes 1-3, then it limits what <code>_DesignName</code> returns to the contents of the folder or room given by the 2<sup>nd</sup> argument.</p> <p>The optional 3<sup>rd</sup> argument limits what <code>_DesignName</code> returns to those objects that have that tag. If you are using the 3<sup>rd</sup> or 4<sup>th</sup> argument and don't want to limit to a specific room, use "" for the 2<sup>nd</sup> argument.</p> <p>If the optional 4<sup>th</sup> argument is used, it is further limited to those objects that have the tag with the given value.</p> <p>A valid return is a number from 0 to 15 inclusive.</p> |
| <b>Example</b>     | <code>hDesign = _DesignOpen(1, "Kitchen")</code>   |

|                    |   |
|--------------------|---|
| <b>Name</b>        | DesignClose   |
| <b>Result type</b> | Void  |
| <b>Parameters</b>  | (# returned from DesignOpen)  |
| <b>Action</b>      | Close the design opened with _DesignOpen. The one argument must be the value returned from _DesignOpen. |
| <b>Example</b>     | Void = _DesignClose (hDesign)   |

|                    |  |
|--------------------|--|
| <b>Name</b>        | DesignName   |
| <b>Result type</b> | String   |
| <b>Parameters</b>  | (# returned from DesignOpen)   |
| <b>Action</b>      | Returns the name of the current design element and moves to the next element. This allows you to use _DesignName until it returns an empty string. This generates the names of all your design elements based upon the arguments to _DesignOpen. |
| <b>Example</b>     | Name = _DesignName (hDesign)   |

|                    |  |
|--------------------|--|
| <b>Name</b>        | DesignTitle  |
| <b>Result type</b> | String   |
| <b>Parameters</b>  | None   |
| <b>Action</b>      | Returns the title set in the Home Properties dialog. |
| <b>Example</b>     | _DesignTitle() → “Kimberly’s Villa”                  |

|                    |   |
|--------------------|---|
| <b>Name</b>        | DeviceForAddress  |
| <b>Result type</b> | String  |
| <b>Parameters</b>  | (“Protocol name”, “address”)  |
| <b>Action</b>      | Locates a device and returns the name of that device by looking for a device of the supplied protocol with the address. The address is formatted differently for each protocol. The protocols are: “X10”, “Insteon”, “UPB”, “Hue”, “Wireless”, or the name of a user class. Returns the empty string if no such device. |
| <b>Example</b>     | _DeviceForAddress(“Insteon”, “02.62.4b”) → “Den – Light”.   |

|                    |   |
|--------------------|---|
| <b>Name</b>        | GetDeviceKind   |
| <b>Result type</b> | Number  |
| <b>Parameters</b>  | (“device name”)   |
| <b>Action</b>      | Returns a value that is the kind of device. possibilities are:<br>0: Other, 1: Switch, 2: Module, 3: Light, 4: Input, 5: Lock, 6: Camera, 7: Keypad with load, 8: Keypad, 9: IR Output, 10: Fan, 11: Thermostat |
| <b>Example</b>     | _GetDeviceKind (“Kitchen-Lights”) → 1   |



|                    |  |
|--------------------|--|
| <b>Name</b>        | GetDeviceMake  |
| <b>Result type</b> | String   |
| <b>Parameters</b>  | ("device name")                                      |
| <b>Action</b>      | Returns the name of the device manufacturer if known |
| <b>Example</b>     | _GetDeviceMake ("Kitchen – lights") → "PulseWorx"    |

|                    |   |
|--------------------|---|
| <b>Name</b>        | GetDeviceModel  |
| <b>Result type</b> | String  |
| <b>Parameters</b>  | ("device name")   |
| <b>Action</b>      | Returns the name of the device model if known             |
| <b>Example</b>     | _GetDeviceModel ("Kitchen – lights") → "WS1D Wall switch" |

|                    |  |
|--------------------|--|
| <b>Name</b>        | GetDeviceProtocol  |
| <b>Result type</b> | String   |
| <b>Parameters</b>  | ("device name")  |
| <b>Action</b>      | Returns the protocol of the device. The protocols are: "X10", "Insteon", "UPB", "Hue", "Wireless" or the name of a user class. |
| <b>Example</b>     | _GetDeviceProtocol ("Kitchen – lights") → "UPB"  |

|                    |   |
|--------------------|---|
| <b>Name</b>        | GroupMemberCount                            |
| <b>Result type</b> | Number                                      |
| <b>Parameters</b>  | ("group name")                              |
| <b>Action</b>      | Returns the number of members in the group. |
| <b>Example</b>     | _GroupMemberCount ("Outside – lights") → 5  |

|                    |   |
|--------------------|---|
| <b>Name</b>        | GroupMemberName   |
| <b>Result type</b> | String  |
| <b>Parameters</b>  | ("group name", member#)                                 |
| <b>Action</b>      | Returns the name of the ith member of the group.        |
| <b>Example</b>     | _GroupMemberName ("Outside – lights", 2) → "Left light" |

|                    |  |
|--------------------|--|
| <b>Name</b>        | HomeMode   |
| <b>Result type</b> | Number   |
| <b>Parameters</b>  | None   |
| <b>Action</b>      | Returns current home mode.<br>0 = Home & Awake, 1 = Home & Asleep, 2 = Away, 3 = 4 <sup>th</sup> mode. |
| <b>Example</b>     | _HomeMode() → 0  |

|                    |  |
|--------------------|--|
| <b>Name</b>        | IsValidObject  |
| <b>Result type</b> | Bool   |
| <b>Parameters</b>  | ("name", code#)  |
| <b>Action</b>      | Returns YES if the design contains an object with the name of the specified type. Codes are:<br>0:Device, 1:Program, 2:Group, 3:Room/Folder/Display, 4:Schedule, 5:Global Variable, 6:room, 7:folder, 8:display. |
| <b>Example</b>     | _IsValidObject("Kitchen – lights", 0) → yes  |

|                    |   |
|--------------------|---|
| <b>Name</b>        | LastControlTime   |
| <b>Result type</b> | Date-Time   |
| <b>Parameters</b>  | ("device, program, or room name")   |
| <b>Action</b>      | Returns the time of the last control of the device or, for a program, when last started. Returns a Boolean NO, if never controlled so use _IsBool to check the result. If a room name is supplied, it returns the latest control of any device in the room. |
| <b>Example</b>     | _LastControlTime ("Kitchen – Lights") → 28-Sep-1018 07:03   |

|                    |   |
|--------------------|---|
| <b>Name</b>        | LastReceptionTime   |
| <b>Result type</b> | Date-time   |
| <b>Parameters</b>  | ("device or room name")   |
| <b>Action</b>      | Returns the time of the last reception from the device. Returns a Boolean NO, if never received from so use _IsBool to check the result. If a room name is supplied, it returns the latest reception from any device in the room. |
| <b>Example</b>     | _LastReceptionTime ("Kitchen – Lights") → 28-Sep-1018 07:04   |

|                    |   |
|--------------------|---|
| <b>Name</b>        | ObjectTagClear  |
| <b>Result type</b> | Bool  |
| <b>Parameters</b>  | ("object name")   |
| <b>Action</b>      | Removes all tags from a device, program, group, room, folder, or display. |
| <b>Example</b>     | _ObjectTagClear ("Kitchen – Lights")                                      |

|                    |  |
|--------------------|--|
| <b>Name</b>        | ObjectTagDelete  |
| <b>Result type</b> | Bool   |
| <b>Parameters</b>  | ("object name", "tag name")  |
| <b>Action</b>      | Removes from a device, program, group, room, folder, or display the supplied tag. If the tag doesn't exist for that object, NO is returned, YES otherwise. |
| <b>Example</b>     | _ObjectTagDelete ("Kitchen – Lights", "color") → yes   |

|                    |  |
|--------------------|--|
| <b>Name</b>        | ObjectTagExists  |
| <b>Result type</b> | Bool   |
| <b>Parameters</b>  | ("object name", "tag name", ["tag value"])   |
| <b>Action</b>      | <p>Checks if a device, program, group, room, folder, or display has the supplied tag and optionally checks that the tag value matches the supplied value.</p> <p>If the tag isn't assigned to the object, NO is returned.</p> <p>If the tag is assigned to the object and the value argument is omitted, YES is returned.</p> <p>If the value argument is supplied, YES if returned if the tag value matches the supplied value.</p> |
| <b>Example</b>     | _ObjectTagExists("Kitchen – Lights", "color", "blue") → yes  |

|                    |  |
|--------------------|--|
| <b>Name</b>        | ObjectTagGet   |
| <b>Result type</b> | String / Bool  |
| <b>Parameters</b>  | ("object name", "tag name")  |
| <b>Action</b>      | Returns the value of the tag assigned to the device, program, group, room, folder, or display with the supplied name. If the object doesn't have that tag returns NO. Use _IsBool to check the result. |
| <b>Example</b>     | _ObjectTagGet("Kitchen – lights", "color") → "blue"  |

|                    |   |
|--------------------|---|
| <b>Name</b>        | ObjectTagSet  |
| <b>Result type</b> | Bool  |
| <b>Parameters</b>  | ("object name", "tag name", "tag value")  |
| <b>Action</b>      | <p>Assigns to the supplied device, program, group, room, folder, or display a tag with the supplied value. If the tag doesn't exist for that object one is added to it.</p> <p>If there is no room for a new tag for that object - it already has the maximum number - a bool NO is returned.</p> |
| <b>Example</b>     | _ObjectTagSet("Kitchen – lights", "color", "blue") → yes  |

|                    |   |
|--------------------|---|
| <b>Name</b>        | SetCurrentWattage   |
| <b>Result type</b> | Void  |
| <b>Parameters</b>  | ("device name", wattage#)   |
| <b>Action</b>      | Change the current wattage used by the device when at 100% to the value supplied. |
| <b>Example</b>     | _SetCurrentWattage("Kitchen – lights", 400)                                       |

|                    |   |
|--------------------|---|
| <b>Name</b>        | Statistics  |
| <b>Result type</b> | Number  |
| <b>Parameters</b>  | (code#, ["name"])   |
| <b>Action</b>      | <p>Returns statistics since HCA was started based upon the supplied code.</p> <p>[Code = 1] Total number devices controlled, unless a device name is given and then the count is for only that device</p> <p>[Code = 2] Total number of programs executed, unless a program name is given and then the count is for only that program</p> <p>[Code = 3] Total number of client connections</p> <p>[Code = 4] Total number of messages from all interfaces unless the interface number is given then the count is for only that interface. Interfaces are numbered 1 to 8. Note that is different than <code>_InterfaceName</code> and <code>_InterfaceStatus</code>.</p> <p>If the code has a negative value it clears the statistics, either the total or for a specific object. Examples:</p> <p><code>_Statistics (1)</code> → the total count for all devices<br/> <code>_Statistics (1, "Kitchen")</code> → the total for all devices in the kitchen<br/> <code>_Statistics (1, "Kitchen - Lights")</code> → the count for only that single device<br/> <code>_Statistics (-1)</code> → clear counter for every device<br/> <code>_Statistics (-1, "Kitchen")</code> → clear counter for every device in the kitchen<br/> <code>_Statistics (-1, "Kitchen - Lights")</code> → clear counter for kitchen-Lights</p> |
| <b>Example</b>     | <code>_Statistics(2)</code> → 26  |

|                    |  |
|--------------------|--|
| <b>Name</b>        | Status   |
| <b>Result type</b> | String   |
| <b>Parameters</b>  | ("object name")  |
| <b>Action</b>      | <p>Returns a readable string which shows the status of the device, program, or group. What is returned depends upon the type of the object, and if a device, the type of the device.</p> |
| <b>Example</b>     | <code>_Status("Kitchen - lights")</code> → "75%"   |

---

## Miscellaneous functions

Functions comprises a set of generally useful things that are not in any other category.

| <b>Name</b>                           | AlertAdd  |                                    |                 |     |      |      |      |            |        |                 |          |                        |         |             |                 |                          |    |             |                 |                                |    |             |                 |                                      |    |             |                 |                             |    |             |                 |                                       |    |             |                 |                   |    |                 |          |               |    |              |                 |                 |    |                                    |          |                      |    |                |          |           |    |                     |          |                |    |                     |          |                            |    |                       |          |                   |    |                 |          |                          |    |                 |          |                  |    |                 |          |
|---------------------------------------|---|------------------------------------|-----------------|-----|------|------|------|------------|--------|-----------------|----------|------------------------|---------|-------------|-----------------|--------------------------|----|-------------|-----------------|--------------------------------|----|-------------|-----------------|--------------------------------------|----|-------------|-----------------|-----------------------------|----|-------------|-----------------|---------------------------------------|----|-------------|-----------------|-------------------|----|-----------------|----------|---------------|----|--------------|-----------------|-----------------|----|------------------------------------|----------|----------------------|----|----------------|----------|-----------|----|---------------------|----------|----------------|----|---------------------|----------|----------------------------|----|-----------------------|----------|-------------------|----|-----------------|----------|--------------------------|----|-----------------|----------|------------------|----|-----------------|----------|
| <b>Result type</b>                    | Void  |                                    |                 |     |      |      |      |            |        |                 |          |                        |         |             |                 |                          |    |             |                 |                                |    |             |                 |                                      |    |             |                 |                             |    |             |                 |                                       |    |             |                 |                   |    |                 |          |               |    |              |                 |                 |    |                                    |          |                      |    |                |          |           |    |                     |          |                |    |                     |          |                            |    |                       |          |                   |    |                 |          |                          |    |                 |          |                  |    |                 |          |
| <b>Parameters</b>                     | (alert#, "arg1", "arg2")  |                                    |                 |     |      |      |      |            |        |                 |          |                        |         |             |                 |                          |    |             |                 |                                |    |             |                 |                                      |    |             |                 |                             |    |             |                 |                                       |    |             |                 |                   |    |                 |          |               |    |              |                 |                 |    |                                    |          |                      |    |                |          |           |    |                     |          |                |    |                     |          |                            |    |                       |          |                   |    |                 |          |                          |    |                 |          |                  |    |                 |          |
| <b>Action</b>                         | <p>Raises an alert. How alerts are configured determines the effect of this. Any of the alerts in HCA can be added. The following table shows the codes and the expected text of the arguments.</p> <table border="1"> <thead> <tr> <th>Use</th> <th>Code</th> <th>Arg1</th> <th>Arg2</th> </tr> </thead> <tbody> <tr> <td>User 1 - 8</td> <td>1 to 8</td> <td>Any text wanted</td> <td>Not used</td> </tr> <tr> <td>No reception group 1-4</td> <td>9 to 12</td> <td>Device name</td> <td>Any text wanted</td> </tr> <tr> <td>Confirm receipt ACK fail</td> <td>13</td> <td>Device name</td> <td>Any text wanted</td> </tr> <tr> <td>Confirm receipt by status fail</td> <td>14</td> <td>Device name</td> <td>Any text wanted</td> </tr> <tr> <td>Confirm receipt by status all failed</td> <td>15</td> <td>Device name</td> <td>Any text wanted</td> </tr> <tr> <td>Status request not answered</td> <td>16</td> <td>Device name</td> <td>Any text wanted</td> </tr> <tr> <td>UPB reception missing sequence packet</td> <td>17</td> <td>Device name</td> <td>Any text wanted</td> </tr> <tr> <td>Unknown reception</td> <td>18</td> <td>Any text wanted</td> <td>Not used</td> </tr> <tr> <td>Program error</td> <td>19</td> <td>Program name</td> <td>Any text wanted</td> </tr> <tr> <td>Interface error</td> <td>20</td> <td>Interface name + ": " + error text</td> <td>Not used</td> </tr> <tr> <td>Interface disconnect</td> <td>21</td> <td>Interface name</td> <td>Not used</td> </tr> <tr> <td>Power out</td> <td>22</td> <td>String showing time</td> <td>Not used</td> </tr> <tr> <td>Power restored</td> <td>23</td> <td>String showing time</td> <td>Not used</td> </tr> <tr> <td>Weather observation failed</td> <td>24</td> <td>Weather provider name</td> <td>Not used</td> </tr> <tr> <td>Client disconnect</td> <td>25</td> <td>Any text wanted</td> <td>Not used</td> </tr> <tr> <td>Cloud DDNS update failed</td> <td>26</td> <td>Any text wanted</td> <td>Not used</td> </tr> <tr> <td>Cloud disconnect</td> <td>27</td> <td>Any text wanted</td> <td>Not used</td> </tr> </tbody> </table> |                                    |                 | Use | Code | Arg1 | Arg2 | User 1 - 8 | 1 to 8 | Any text wanted | Not used | No reception group 1-4 | 9 to 12 | Device name | Any text wanted | Confirm receipt ACK fail | 13 | Device name | Any text wanted | Confirm receipt by status fail | 14 | Device name | Any text wanted | Confirm receipt by status all failed | 15 | Device name | Any text wanted | Status request not answered | 16 | Device name | Any text wanted | UPB reception missing sequence packet | 17 | Device name | Any text wanted | Unknown reception | 18 | Any text wanted | Not used | Program error | 19 | Program name | Any text wanted | Interface error | 20 | Interface name + ": " + error text | Not used | Interface disconnect | 21 | Interface name | Not used | Power out | 22 | String showing time | Not used | Power restored | 23 | String showing time | Not used | Weather observation failed | 24 | Weather provider name | Not used | Client disconnect | 25 | Any text wanted | Not used | Cloud DDNS update failed | 26 | Any text wanted | Not used | Cloud disconnect | 27 | Any text wanted | Not used |
| Use                                   | Code  | Arg1                               | Arg2            |     |      |      |      |            |        |                 |          |                        |         |             |                 |                          |    |             |                 |                                |    |             |                 |                                      |    |             |                 |                             |    |             |                 |                                       |    |             |                 |                   |    |                 |          |               |    |              |                 |                 |    |                                    |          |                      |    |                |          |           |    |                     |          |                |    |                     |          |                            |    |                       |          |                   |    |                 |          |                          |    |                 |          |                  |    |                 |          |
| User 1 - 8                            | 1 to 8  | Any text wanted                    | Not used        |     |      |      |      |            |        |                 |          |                        |         |             |                 |                          |    |             |                 |                                |    |             |                 |                                      |    |             |                 |                             |    |             |                 |                                       |    |             |                 |                   |    |                 |          |               |    |              |                 |                 |    |                                    |          |                      |    |                |          |           |    |                     |          |                |    |                     |          |                            |    |                       |          |                   |    |                 |          |                          |    |                 |          |                  |    |                 |          |
| No reception group 1-4                | 9 to 12   | Device name                        | Any text wanted |     |      |      |      |            |        |                 |          |                        |         |             |                 |                          |    |             |                 |                                |    |             |                 |                                      |    |             |                 |                             |    |             |                 |                                       |    |             |                 |                   |    |                 |          |               |    |              |                 |                 |    |                                    |          |                      |    |                |          |           |    |                     |          |                |    |                     |          |                            |    |                       |          |                   |    |                 |          |                          |    |                 |          |                  |    |                 |          |
| Confirm receipt ACK fail              | 13  | Device name                        | Any text wanted |     |      |      |      |            |        |                 |          |                        |         |             |                 |                          |    |             |                 |                                |    |             |                 |                                      |    |             |                 |                             |    |             |                 |                                       |    |             |                 |                   |    |                 |          |               |    |              |                 |                 |    |                                    |          |                      |    |                |          |           |    |                     |          |                |    |                     |          |                            |    |                       |          |                   |    |                 |          |                          |    |                 |          |                  |    |                 |          |
| Confirm receipt by status fail        | 14  | Device name                        | Any text wanted |     |      |      |      |            |        |                 |          |                        |         |             |                 |                          |    |             |                 |                                |    |             |                 |                                      |    |             |                 |                             |    |             |                 |                                       |    |             |                 |                   |    |                 |          |               |    |              |                 |                 |    |                                    |          |                      |    |                |          |           |    |                     |          |                |    |                     |          |                            |    |                       |          |                   |    |                 |          |                          |    |                 |          |                  |    |                 |          |
| Confirm receipt by status all failed  | 15  | Device name                        | Any text wanted |     |      |      |      |            |        |                 |          |                        |         |             |                 |                          |    |             |                 |                                |    |             |                 |                                      |    |             |                 |                             |    |             |                 |                                       |    |             |                 |                   |    |                 |          |               |    |              |                 |                 |    |                                    |          |                      |    |                |          |           |    |                     |          |                |    |                     |          |                            |    |                       |          |                   |    |                 |          |                          |    |                 |          |                  |    |                 |          |
| Status request not answered           | 16  | Device name                        | Any text wanted |     |      |      |      |            |        |                 |          |                        |         |             |                 |                          |    |             |                 |                                |    |             |                 |                                      |    |             |                 |                             |    |             |                 |                                       |    |             |                 |                   |    |                 |          |               |    |              |                 |                 |    |                                    |          |                      |    |                |          |           |    |                     |          |                |    |                     |          |                            |    |                       |          |                   |    |                 |          |                          |    |                 |          |                  |    |                 |          |
| UPB reception missing sequence packet | 17  | Device name                        | Any text wanted |     |      |      |      |            |        |                 |          |                        |         |             |                 |                          |    |             |                 |                                |    |             |                 |                                      |    |             |                 |                             |    |             |                 |                                       |    |             |                 |                   |    |                 |          |               |    |              |                 |                 |    |                                    |          |                      |    |                |          |           |    |                     |          |                |    |                     |          |                            |    |                       |          |                   |    |                 |          |                          |    |                 |          |                  |    |                 |          |
| Unknown reception                     | 18  | Any text wanted                    | Not used        |     |      |      |      |            |        |                 |          |                        |         |             |                 |                          |    |             |                 |                                |    |             |                 |                                      |    |             |                 |                             |    |             |                 |                                       |    |             |                 |                   |    |                 |          |               |    |              |                 |                 |    |                                    |          |                      |    |                |          |           |    |                     |          |                |    |                     |          |                            |    |                       |          |                   |    |                 |          |                          |    |                 |          |                  |    |                 |          |
| Program error                         | 19  | Program name                       | Any text wanted |     |      |      |      |            |        |                 |          |                        |         |             |                 |                          |    |             |                 |                                |    |             |                 |                                      |    |             |                 |                             |    |             |                 |                                       |    |             |                 |                   |    |                 |          |               |    |              |                 |                 |    |                                    |          |                      |    |                |          |           |    |                     |          |                |    |                     |          |                            |    |                       |          |                   |    |                 |          |                          |    |                 |          |                  |    |                 |          |
| Interface error                       | 20  | Interface name + ": " + error text | Not used        |     |      |      |      |            |        |                 |          |                        |         |             |                 |                          |    |             |                 |                                |    |             |                 |                                      |    |             |                 |                             |    |             |                 |                                       |    |             |                 |                   |    |                 |          |               |    |              |                 |                 |    |                                    |          |                      |    |                |          |           |    |                     |          |                |    |                     |          |                            |    |                       |          |                   |    |                 |          |                          |    |                 |          |                  |    |                 |          |
| Interface disconnect                  | 21  | Interface name                     | Not used        |     |      |      |      |            |        |                 |          |                        |         |             |                 |                          |    |             |                 |                                |    |             |                 |                                      |    |             |                 |                             |    |             |                 |                                       |    |             |                 |                   |    |                 |          |               |    |              |                 |                 |    |                                    |          |                      |    |                |          |           |    |                     |          |                |    |                     |          |                            |    |                       |          |                   |    |                 |          |                          |    |                 |          |                  |    |                 |          |
| Power out                             | 22  | String showing time                | Not used        |     |      |      |      |            |        |                 |          |                        |         |             |                 |                          |    |             |                 |                                |    |             |                 |                                      |    |             |                 |                             |    |             |                 |                                       |    |             |                 |                   |    |                 |          |               |    |              |                 |                 |    |                                    |          |                      |    |                |          |           |    |                     |          |                |    |                     |          |                            |    |                       |          |                   |    |                 |          |                          |    |                 |          |                  |    |                 |          |
| Power restored                        | 23  | String showing time                | Not used        |     |      |      |      |            |        |                 |          |                        |         |             |                 |                          |    |             |                 |                                |    |             |                 |                                      |    |             |                 |                             |    |             |                 |                                       |    |             |                 |                   |    |                 |          |               |    |              |                 |                 |    |                                    |          |                      |    |                |          |           |    |                     |          |                |    |                     |          |                            |    |                       |          |                   |    |                 |          |                          |    |                 |          |                  |    |                 |          |
| Weather observation failed            | 24  | Weather provider name              | Not used        |     |      |      |      |            |        |                 |          |                        |         |             |                 |                          |    |             |                 |                                |    |             |                 |                                      |    |             |                 |                             |    |             |                 |                                       |    |             |                 |                   |    |                 |          |               |    |              |                 |                 |    |                                    |          |                      |    |                |          |           |    |                     |          |                |    |                     |          |                            |    |                       |          |                   |    |                 |          |                          |    |                 |          |                  |    |                 |          |
| Client disconnect                     | 25  | Any text wanted                    | Not used        |     |      |      |      |            |        |                 |          |                        |         |             |                 |                          |    |             |                 |                                |    |             |                 |                                      |    |             |                 |                             |    |             |                 |                                       |    |             |                 |                   |    |                 |          |               |    |              |                 |                 |    |                                    |          |                      |    |                |          |           |    |                     |          |                |    |                     |          |                            |    |                       |          |                   |    |                 |          |                          |    |                 |          |                  |    |                 |          |
| Cloud DDNS update failed              | 26  | Any text wanted                    | Not used        |     |      |      |      |            |        |                 |          |                        |         |             |                 |                          |    |             |                 |                                |    |             |                 |                                      |    |             |                 |                             |    |             |                 |                                       |    |             |                 |                   |    |                 |          |               |    |              |                 |                 |    |                                    |          |                      |    |                |          |           |    |                     |          |                |    |                     |          |                            |    |                       |          |                   |    |                 |          |                          |    |                 |          |                  |    |                 |          |
| Cloud disconnect                      | 27  | Any text wanted                    | Not used        |     |      |      |      |            |        |                 |          |                        |         |             |                 |                          |    |             |                 |                                |    |             |                 |                                      |    |             |                 |                             |    |             |                 |                                       |    |             |                 |                   |    |                 |          |               |    |              |                 |                 |    |                                    |          |                      |    |                |          |           |    |                     |          |                |    |                     |          |                            |    |                       |          |                   |    |                 |          |                          |    |                 |          |                  |    |                 |          |
| <b>Example</b>                        | _AlertAdd(1, "Check pump")  |                                    |                 |     |      |      |      |            |        |                 |          |                        |         |             |                 |                          |    |             |                 |                                |    |             |                 |                                      |    |             |                 |                             |    |             |                 |                                       |    |             |                 |                   |    |                 |          |               |    |              |                 |                 |    |                                    |          |                      |    |                |          |           |    |                     |          |                |    |                     |          |                            |    |                       |          |                   |    |                 |          |                          |    |                 |          |                  |    |                 |          |

|                    |   |
|--------------------|---|
| <b>Name</b>        | AlertCount  |
| <b>Result type</b> | Number  |
| <b>Parameters</b>  | (alert#, [device or room name])   |
| <b>Action</b>      | <p>Return the number of alerts for the supplied code. For alerts related to a device, if the optional second argument is supplied as a device name, then the count of alerts of the type for that device are returned. If a room name is supplied, then a total count of alerts for all device in the room with that type of alert,</p> <p>Codes are:</p> <ul style="list-style-type: none"> <li>1: User alert 1</li> <li>2: User alert 2</li> <li>3: User alert 3</li> <li>4: User alert 4</li> <li>5: User alert 5</li> <li>6: User alert 6</li> <li>7: User alert 7</li> <li>8: User alert 8</li> <li>9: No reception group 1</li> <li>10: No reception group 2</li> <li>11: No reception group 3</li> <li>12: No reception group 4</li> <li>13: No ACK from device (Confirm receipt failed)</li> <li>14: Confirm receipt of command failed</li> <li>15: All attempts at confirm receipt of command failed</li> <li>16: Status poll fails</li> <li>17: UPB missing sequence packet</li> <li>18: Unknown reception</li> <li>19: Program error</li> <li>20: Interface error</li> <li>21: Interface disconnect</li> <li>22: Power out (interface disconnected)</li> <li>23: Power restored (interface reconnected)</li> <li>24: Weather observation failed</li> <li>25: Client disconnected abnormally</li> <li>26: Cloud update failed</li> <li>27: Cloud connected server disconnected</li> </ul> |
| <b>Example</b>     | <code>_AlertCount(7) → 1</code>   |

|                    |   |
|--------------------|---|
| <b>Name</b>        | AlertClear  |
| <b>Result type</b> | Void  |
| <b>Parameters</b>  | (alert#, [device or room name])   |
| <b>Action</b>      | <p>Clear the alert given by the #. The Alert number is the same as supplied to <code>_AlertCount</code>. If the optional second argument is supplied as a device name, then that alert for only that device is cleared. If a room name is supplied, then that alert type for all devices in the room are cleared.</p> |
| <b>Example</b>     | <code>_AlertClear(11)</code>  |

|                    |   |
|--------------------|---|
| <b>Name</b>        | AlertLevel  |
| <b>Result type</b> | Number  |
| <b>Parameters</b>  | (alert#)  |
| <b>Action</b>      | Returns the level of the alert. This is the color for that alert as seen in the Alert Manager. Green=0, Yellow=1, Red=2 |
| <b>Example</b>     | _AlertLevel(11) → 1   |

|                    |   |
|--------------------|---|
| <b>Name</b>        | Assign  |
| <b>Result type</b> | Void  |
| <b>Parameters</b>  | ("variable name", any)  |
| <b>Action</b>      | Assigns to the named variable – given by a string - the value given by the second argument. If the variable doesn't exist a global variable is created. |
| <b>Example</b>     | _Assign("My variable", 17)  |

|                    |  |
|--------------------|--|
| <b>Name</b>        | Delay  |
| <b>Result type</b> | Number   |
| <b>Parameters</b>  | (time-span1, [time-span2])   |
| <b>Action</b>      | If one argument is supplied, then delays for that amount of time. If two arguments are supplied, then delays for a time somewhere between the two time spans. Returns the number of seconds delayed. |
| <b>Example</b>     | _Delay(_Minutes(1), _Minutes(10)) → 117  |

|                    |   |
|--------------------|---|
| <b>Name</b>        | DelayShort  |
| <b>Result type</b> | Void  |
| <b>Parameters</b>  | (milliseconds#)   |
| <b>Action</b>      | Delays for at least the specified number of milliseconds. |
| <b>Example</b>     | _DelayShort(1500)   |

|                    |  |
|--------------------|--|
| <b>Name</b>        | DesignSave   |
| <b>Result type</b> | Void   |
| <b>Parameters</b>  | ([code#])  |
| <b>Action</b>      | Saves the current design file. The codes are: 0 or not supplied = Always save.<br>1 = Only save if modified. |
| <b>Example</b>     | _DesignSave ()   |

|                    |  |
|--------------------|--|
| <b>Name</b>        | GetFolder  |
| <b>Result type</b> | String   |
| <b>Parameters</b>  | ("2-part name")                                    |
| <b>Action</b>      | Returns the folder name portion of the 2-part name |
| <b>Example</b>     | _GetFolder("Kitchen – Lights") → "Kitchen"         |

|                    |   |
|--------------------|---|
| <b>Name</b>        | GetObject                                   |
| <b>Result type</b> | String                                      |
| <b>Parameters</b>  | (2"2-part name")                            |
| <b>Action</b>      | Returns the object portion f a 2-part name. |
| <b>Example</b>     | _GetFolder("Kitchen – Lights") → "Lights"   |

|                    |  |
|--------------------|--|
| <b>Name</b>        | HCAVersionGE   |
| <b>Result type</b> | Bool   |
| <b>Parameters</b>  | (majorVersion, [minorVersion], [Build#])   |
| <b>Action</b>      | Generates a yes/no value based upon the version of HCA running the program and the supplied arguments.   |
| <b>Example</b>     | <p>Example, assume that the current HCA version that is running the program is 16.0.28</p> <pre> _HCAVersionGE(16) -&gt; Yes _HCAVersionGE(17) -&gt; No _HCAVersionGE(16, 0) -&gt; Yes _HCAVersionGE(16, 1) -&gt; No _HCAVersionGE(16, 0, 28) -&gt; Yes _HCAVersionGE(16, 0, 29) -&gt; No _HCAVersionGE(16, 1, 30) -&gt; No </pre> |



|                    |   |
|--------------------|---|
| <b>Name</b>        | HostNameToIP  |
| <b>Result type</b> | String  |
| <b>Parameters</b>  | (host name string)  |
| <b>Action</b>      | Uses the name server to convert a host name to the 4-part IP address. Returns a Boolean NO if the conversion fails. |
| <b>Example</b>     | <code>_HostNameToIP("hcatech.com")</code> → "66.113.102.195"  |

|                    |  |
|--------------------|--|
| <b>Name</b>        | InterfaceName  |
| <b>Result type</b> | String   |
| <b>Parameters</b>  | (interface-number)   |
| <b>Action</b>      | Returns the name of the interface with the supplied number (0-7). The numbers are in the order that the interfaces appear on the HCA Options hardware tab. |
| <b>Example</b>     | <code>_InterfaceName()</code> → "Insteon"  |

|                    |   |
|--------------------|---|
| <b>Name</b>        | InterfaceRestart  |
| <b>Result type</b> | Bool  |
| <b>Parameters</b>  | None  |
| <b>Action</b>      | Disconnects HCA from all automation interfaces and then reconnects. This can help keep the interfaces connected over a long period of time in case Windows decided to close connections not recently used. Returns YES if the operation worked. |
| <b>Example</b>     | <code>_InterfaceRestart()</code> → TRUE   |

|                    |   |
|--------------------|---|
| <b>Name</b>        | InterfaceStatus   |
| <b>Result type</b> | Bool  |
| <b>Parameters</b>  | (interface-number)  |
| <b>Action</b>      | Returns the status of the interface with the supplied number (0-7). The numbers are in the order that the interfaces appear on the HCA Options hardware tab. Returns TRUE if working and FALSE if not |
| <b>Example</b>     | <code>_InterfaceStatus(1)</code> → TRUE   |

|                    |  |
|--------------------|--|
| <b>Name</b>        | MakeValidName  |
| <b>Result type</b> | string   |
| <b>Parameters</b>  | ("string")   |
| <b>Action</b>      | Converts if necessary, the supplied string into a valid HCA Name. Any characters in a HCA name that are not allowed are replaced by underscore characters. |
| <b>Example</b>     | <code>_MakeValidName("Light:1")</code> → "Light_1"   |

|                    |  |
|--------------------|--|
| <b>Name</b>        | PgmNote  |
| <b>Result type</b> | void   |
| <b>Parameters</b>  | ("string", [code])   |
| <b>Action</b>      | Displays the text in the program note window if open. If the program notes window is not open, has no effect. The optional "code" argument is not used at this time. |
| <b>Example</b>     | _PgmNote("Handling change in levels")  |

|                    |   |
|--------------------|---|
| <b>Name</b>        | PlaySound   |
| <b>Result type</b> | Bool  |
| <b>Parameters</b>  | ("path to sound file", code#)   |
| <b>Action</b>      | <p>Plays a Sound file using the computers sound system. The 1st argument is a path to the sound file.</p> <p>The code# argument is as follows:</p> <ol style="list-style-type: none"> <li>1: The sound file starts playing and HCA moves to the next element</li> <li>2: The sound file starts playing and HCA moves to the next element. When the sound file finishes, it starts playing again.</li> <li>3: The sound file starts playing and HCA waits until it is complete before moving to the next element.</li> </ol> <p>If you used option #2, later you can stop the sound file playing by using the PlaySound function again with "" for the path.</p> |
| <b>Example</b>     | _PlaySound ("c:\files\beep.wav", 1) → yes   |

|                    |   |
|--------------------|---|
| <b>Name</b>        | ProblemLevel  |
| <b>Result type</b> | Number  |
| <b>Parameters</b>  | None  |
| <b>Action</b>      | Returns current problem level as shown by the HCA status bar lights.<br>0 = Green, 1 = Yellow, 2 = Red. |
| <b>Example</b>     | _ProblemLevel() → 0   |

|                    |  |
|--------------------|--|
| <b>Name</b>        | RGB  |
| <b>Result type</b> | Number   |
| <b>Parameters</b>  | (red#, blue#, green#)  |
| <b>Action</b>      | Returns the encoded color for the red, green, blue values specified. |
| <b>Example</b>     | _RGB(51, 51,255) → hex value 3333FF (a nice blue)                    |

|                    |  |
|--------------------|--|
| <b>Name</b>        | Rand   |
| <b>Result type</b> | Number   |
| <b>Parameters</b>  | (number1, number2)   |
| <b>Action</b>      | Returns a random number chosen between the two numbers supplied. |
| <b>Example</b>     | _Rand (100, 200) → 119   |

|                    |   |
|--------------------|---|
| <b>Name</b>        | ReportAdd   |
| <b>Result type</b> | Void  |
| <b>Parameters</b>  | ("text")  |
| <b>Action</b>      | Adds a message to be sent in the next Daily Report. |
| <b>Example</b>     | _ReportAdd ("Check battery levels this week")       |

|                    |  |
|--------------------|--|
| <b>Name</b>        | SetSunriseDelta  |
| <b>Result type</b> | Number   |
| <b>Parameters</b>  | (sunrise delta #)  |
| <b>Action</b>      | Changes the sunrise delta value that is set in the home properties on the Location tab. This is the number of minutes to add or subtract from the computed sunrise time to be more accurate for your location. Returns the value it was set to before the change was made. |
| <b>Example</b>     | _SetSunriseDelta (20) → 8  |

|                    |  |
|--------------------|--|
| <b>Name</b>        | SetSunsetDelta   |
| <b>Result type</b> | Number   |
| <b>Parameters</b>  | (sunset delta #)   |
| <b>Action</b>      | Changes the sunset delta value that is set in the home properties on the Location tab. This is the number of minutes to add or subtract from the computed sunset time to be more accurate for your location. Returns the value it was set to before the change was made. |
| <b>Example</b>     | _SetSunsetDelta (20) → 6   |

|                    |   |
|--------------------|---|
| <b>Name</b>        | ThisProgram                                     |
| <b>Result type</b> | String  |
| <b>Parameters</b>  | None  |
| <b>Action</b>      | Returns the 2-part name of the running program. |
| <b>Example</b>     | _ThisProgram() → "Garden - Watering"            |

|                    |   |
|--------------------|---|
| <b>Name</b>        | ThisFolder  |
| <b>Result type</b> | String  |
| <b>Parameters</b>  | None  |
| <b>Action</b>      | Operates like <code>_ThisProgram</code> – to return the name of the running program – but only returns the folder part. |
| <b>Example</b>     | <code>_ThisProgram()</code> → “Garden”  |

| <b>Name</b>        | TileUpdate  |                  |            |      |      |   |              |            |          |   |               |                  |            |   |                   |            |          |   |             |      |          |   |         |          |          |
|--------------------|---|------------------|------------|------|------|---|--------------|------------|----------|---|---------------|------------------|------------|---|-------------------|------------|----------|---|-------------|------|----------|---|---------|----------|----------|
| <b>Result type</b> | Bool  |                  |            |      |      |   |              |            |          |   |               |                  |            |   |                   |            |          |   |             |      |          |   |         |          |          |
| <b>Parameters</b>  | (“tile name”, code#, [x], [y])  |                  |            |      |      |   |              |            |          |   |               |                  |            |   |                   |            |          |   |             |      |          |   |         |          |          |
| <b>Action</b>      | <p>Updates the names tile based upon the parameters</p> <table border="1"> <thead> <tr> <th>Code</th> <th>Use</th> <th>Arg3</th> <th>Arg4</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Change label</td> <td>Label text</td> <td>Not used</td> </tr> <tr> <td>1</td> <td>Change colors</td> <td>Background color</td> <td>Text color</td> </tr> <tr> <td>2</td> <td>Change image path</td> <td>Image path</td> <td>Not used</td> </tr> <tr> <td>3</td> <td>Change text</td> <td>Text</td> <td>Not used</td> </tr> <tr> <td>4</td> <td>Refresh</td> <td>Not used</td> <td>Not used</td> </tr> </tbody> </table> | Code             | Use        | Arg3 | Arg4 | 0 | Change label | Label text | Not used | 1 | Change colors | Background color | Text color | 2 | Change image path | Image path | Not used | 3 | Change text | Text | Not used | 4 | Refresh | Not used | Not used |
| Code               | Use   | Arg3             | Arg4       |      |      |   |              |            |          |   |               |                  |            |   |                   |            |          |   |             |      |          |   |         |          |          |
| 0                  | Change label  | Label text       | Not used   |      |      |   |              |            |          |   |               |                  |            |   |                   |            |          |   |             |      |          |   |         |          |          |
| 1                  | Change colors   | Background color | Text color |      |      |   |              |            |          |   |               |                  |            |   |                   |            |          |   |             |      |          |   |         |          |          |
| 2                  | Change image path   | Image path       | Not used   |      |      |   |              |            |          |   |               |                  |            |   |                   |            |          |   |             |      |          |   |         |          |          |
| 3                  | Change text   | Text             | Not used   |      |      |   |              |            |          |   |               |                  |            |   |                   |            |          |   |             |      |          |   |         |          |          |
| 4                  | Refresh   | Not used         | Not used   |      |      |   |              |            |          |   |               |                  |            |   |                   |            |          |   |             |      |          |   |         |          |          |
| <b>Example</b>     | <code>_TileUpdate (“StatusTile”, 0, “Good”)</code> → yes (if there was a tile with that name, no otherwise).  |                  |            |      |      |   |              |            |          |   |               |                  |            |   |                   |            |          |   |             |      |          |   |         |          |          |

|                    |  |
|--------------------|--|
| <b>Name</b>        | VarValue   |
| <b>Result type</b> | Any  |
| <b>Parameters</b>  | (“variable name”)  |
| <b>Action</b>      | Returns the value of the named variable given as a text string. If the variable doesn't exist a global variable is created with default value of NO. |
| <b>Example</b>     | <code>_VarValue(“Counter”)</code> → 10   |

Abs, 10  
 AddressForDevice, 39  
 AlertAdd, 45  
 AlertClear, 46  
 AlertCount, 46  
 AlertLevel, 47  
 Asc, 7  
 Assign, 47  
 AutoOffTime, 21  
 BarometerConvert, 32  
 BarometerUnits, 32  
 ChangeSchedule, 21  
 Choose, 10  
 Chr, 7  
 CurrentScene, 22  
 CurrentSchedule, 23  
 CurrentWattage, 39  
 DarkSky, 31  
 Date, 14  
 DateTime, 14  
 DateTimeSpan, 17  
 Day, 14  
 DayNight, 23  
 DayOfWeek, 14  
 DayOfYear, 14  
 Days, 15  
 DecToHex, 7  
 Delay, 47  
 DelayShort, 47  
 DesignClose, 40  
 DesignName, 40  
 DesignOpen, 39  
 DesignSave, 47  
 DesignTitle, 40  
 DeviceForAddress, 40  
 DimDownPercent, 23  
 DimPercent, 23  
 DimToPercent, 24  
 DimUpPercent, 24  
 FileClose, 35  
 FileExists, 35  
 FileLoad, 35  
 FileOpen, 35  
 FileReadString, 36  
 FileWriteString, 36  
 FormatInt, 19  
 FormatNum, 19  
 FormatPatten, 19  
 FormatTime, 20  
 GetDeviceKind, 40  
 GetDeviceMake, 41  
 GetDeviceModel, 41  
 GetDeviceProtocol, 41  
 GetFolder, 48  
 GetObject, 48  
 GetThermostat, 30  
 GroupMemberCount, 41  
 GroupMemberName, 41  
 HCAFolder, 36  
 HexToDec, 7  
 HomeMode, 41  
 HostNameToIP, 49  
 Hour, 15  
 Hours, 15  
 Hue, 24  
 HueColorCreate, 25  
 HumidityUnits, 32  
 IconChange, 25  
 IconChangeEx, 25  
 IconExists, 26  
 IIF, 10  
 InsteonBeep, 26  
 InStr, 7  
 Int, 11  
 InterfaceName, 49  
 InterfaceRestart, 49  
 InterfaceStatus, 49  
 IsBool, 11  
 IsCurrentSchedule, 26  
 IsDate, 11  
 IsDim, 26  
 IsDisabled, 26  
 IsEven, 11  
 IsInErrorState, 27  
 IsNumber, 11  
 IsOdd, 12  
 IsOff, 27  
 IsOn, 27  
 IsRunning, 27  
 IsSuspended, 27  
 IsText, 12  
 IsValidObject, 42  
 Json, 37  
 JsonClose, 37  
 JsonCurrentKey, 38  
 JsonCurrentValue, 38  
 JsonDown, 37  
 JsonNext, 38  
 JsonOpen, 37  
 JsonPrev, 38  
 JsonUp, 38  
 LastControlTime, 42

LastReceptionTime, 42  
Lcase, 8  
Left, 8  
Len, 8  
LocalToUTC, 15  
LTrim, 8  
MakeValidName, 49  
Match, 8  
Max, 12  
Mid, 9  
Min, 12  
Minute, 15  
Minutes, 15  
ModifySchedule, 21  
Month, 16  
MonthName, 16  
Now, 16  
Num, 12  
ObjectTagClear, 42  
ObjectTagDelete, 42  
ObjectTagExists, 43  
ObjectTagGet, 43  
ObjectTagSet, 43  
Off, 28  
On, 28  
ParseTime, 16  
PgmNote, 50  
PlaySound, 50  
ProblemLevel, 50  
RainConvert, 32  
RainUnits, 32  
Rand, 51  
ReportAdd, 51  
RGB, 50  
Right, 9  
Round, 13  
RTrim, 9  
Second, 16  
Seconds, 16  
SetCurrentState, 28  
SetCurrentWattage, 43  
SetHomeMode, 28  
SetRunAgainTime, 29  
SetSunriseDelta, 51  
SetSunsetDelta, 51  
SetThermostat, 30  
SetToScene, 29  
SolarRadiationunits, 32  
StartProgram, 29  
Statistics, 44  
Status, 44  
Sunrise, 17  
Sunset, 17  
TempConvert, 33  
TempDecode, 31  
TempUnits, 33  
TextPiece, 9  
TextReplace, 9  
ThisFolder, 52  
ThisProgram, 51  
TileUpdate, 52  
Time, 17  
TimeSpan, 17  
TotalHours, 17  
TotalMinutes, 17  
TotalSeconds, 18  
Trim, 10  
Ucase, 10  
UTCToLocal, 18  
UVUnits, 33  
VarValue, 52  
WeatherGet, 31  
Weekday, 18  
WeekdayName, 18  
WindDirection, 33  
WindDirUnits, 33  
WindSpeedConvert, 33  
WindSpeedUnits, 34  
Year, 18