FeneVision Scripting Reference Guide

Purpose

The goal of this document is to assist the user in writing basic scripts such as attribute scripts within FeneVision.

Getting Started

When you first create an attribute and open the script you will see something similar to this. *Note: It is recommended the first time you open a script window to change the font to 'Courier New'. This is a fixed width font so it makes reading code easier. To do this select 'View' then 'Fonts' at the top of the window.*

```
Function AttributeValue() As Variant

    Dim retval As Variant

    'EXAMPLES
    ' retval = Attributes("Width").Value + 0.5
    ' retval = Parent.Attributes("Width").Value - 4
    ' retval = OptionValue("option")
    ' Include = OptionExists("option")
    ' Include = IsOptionValue("option", value)

    'ENTER YOUR CALCULATION HERE!!
    retval = Parent.Attributes("").value

    'DO NOT MODIFY CODE BELOW THIS LINE
    AttributeValue = retval

End Function
```

The first important note, a single quote (') is used to change a line to a comment. Comment lines appear in green and are used to make notes or describe what the script is doing.

When writing a script you should only make changes between the comment that says 'ENTER YOUR CALCULATION HERE!!' and 'DO NOT MODIFY CODE BELOW THIS LINE'.

```
Function AttributeValue() As Variant

    Dim retval As Variant

    'EXAMPLES
    ' retval = Attributes("Width").Value + 0.5
    ' retval = Parent.Attributes("Width").Value - 4
    ' retval = OptionValue("option")
    ' Include = OptionExists("option")
    ' Include = IsOptionValue("option", value)

    'ENTER YOUR CALCULATION HERE!!
    retval = Parent.Attributes("").value

    'DO NOT MODIFY CODE BELOW THIS LINE
    AttributeValue = retval

End Function
```

Only make changes between these two lines

Variables

The first thing to do in any script is declare any variables that will be needed. The various types of variables take up different amounts of memory, so selecting the variable type carefully can improve the performance of the script. *Note: The varaiable types below are numbered 1 – 6, 1 being the type that takes the least amount of space 6 being the type that takes the most.*

- **Boolean (1) –** Variable that can have only two values, True or False.
- **Integer (2) –** Varaible to store a whole number. This type should not be used if the number might include a fraction or decimal as this only stores whole numbers.
- **Single (3) –** Variable to store a number. The precision of this number will only go out to roughly 4 decimal places. If a greater decimal precision is needed this variable type should not be used.
- **Double (4) –** Variable to store a number with greater decimal precision than 'Single'. This takes roughly double the memory of the 'Single'.
- **String (5) –** Variable to store a sequence of characters. Ex. "ABC", "123", or "AB23".
- **Variant (6) –** Variable that can be any of the other types. This uses the most amount of memory but is the most flexible.

Variables must be declared before they can be used. The syntax to declare a variable is "Dim *Variable Name* As *Variable Type".* Where the 'Variable Name' is user defined and the 'Variable Type is selected from the list above.

If/Else/ElseIf Statements

In many attribute scripts you willl need to return one result if a certain criteria is met and a different result for other criteria. This is accomplished using an 'If/Else' or 'If/ElseIf' statement.

If with Else

First, let's look at how to write and 'If/Else' statement. For example, you might want to say 'If the Width is greater than 40 return that the sash needs reinforcement otherwise return that reinforcement is optional.' This would be scripted with a simple 'If/Else' block as shown below.

```
'Declare the variable to use
Dim W As Single

'Initialize the variable
W = Attributes("W").value

'Start with the IF and check if the W is over 40.
'If W isn't over 40 then it will fall into the ELSE
'Any time IF is used the loop must end with an END IF
If W > 40 Then
    retval = "Sash reinforcement is required."
Else
    retval = "Sash reinforcement is optional."
End If
```

If without Else

Often 'If/Else' statements can be written without the 'Else' part of the statement to simplify scripts. 'Else' is used when the 'If' is not met, you can eliminate that part of the loop by initializing your retval before the 'If'. The example below will do the exact same thing as the 'If/Else' example above.

```
'Declare the variable to use
Dim W As Single

'Initialize the variable
W = Attributes("W").value

'Initialize retval
retval = "Sash reinforcement is optional."

'Use IF and check if Width is over 40.
If W > 40 Then retval = "Sash reinforcement is required."
```

Also, notice that the 'End If' is not necessary when using 'If' without 'Else'. This is ONLY true if the if statement is written on one line.

If with ElseIf

Next let's look at the scenario where you want to return multiple different values based on a specific criteria. For example, if the width is under 40 inches you want it to say reinforcement is optional. If the width is between 40 and 55 you want it to say sash reinforcement is required. If the width is between 55 and 65 you want it to say double sash reinforcement is required but if the Width is over 65 you want it to say the window is too large. The example below illustrates how to script this using a 'If/ElseIf' statement.

```
'Declare the variable to use
Dim W As Single

'Initialize the variable
W = Attributes("W").value

'Use IF and check if the Width is less than or equal to 40
'Next use ElseIf and check if the Width is between 40 and 55
'Next use ElseIf again and check if the Width is between 55 and 65
'Finally use Else to catch anything over 65
If W <= 40 Then
    retval = "Sash reinforcement is optional."
ElseIf W > 40 And W <= 55 Then
    retval = "Sash reinforcement is required."
ElseIf W > 55 And W <= 65 Then
    retval = "Double sash reinforcement is required."
Else
    retval = "Window is too large to be produced."
End If
```

Notice that 'ElseIf' is one word. These is no space. 'End If' is two words. There is a space.

'AND' and 'OR' Statements

Often during scripting you'll need to write an 'AND' or and 'OR' statement. For example, maybe you want to say 'IF the Width is over 40 inches AND the glass is Single Strength THEN a warning should be added'. In order to write statements like this, it is important to understand the results of these statements. This can best be illustrated by the following truth tables. Say 'the Window is over 40 inches' is event A and 'the glass is Single Strength' is event B. The only time that the statement 'IF A AND B' will be True is if A is True and B is true.

If A AND B Then		The statement is
A is True	B is True	True
A is True	B is False	False
A is False	B is True	False
A is False	B is False	False

When using an OR statement, the only time the statement 'IF A OR B' will be False is if A is False and B is False.

If A OR B Then		The statement is
A is True	B is True	True
A is True	B is False	True
A is False	B is True	True
A is False	B is False	False

Examples of 'AND' 'OR' Statements

Fill in the result of each table. We will go over what the result should be.

If (OptionExists("COL") AND OptionExists("LOW E")) OR OptionExists("DIA")

			The statement is
COL is True	LOW E is True	DIA is True	
COL is True	LOW E is True	DIA is False	
COL is True	LOW E is False	DIA is True	
COL is True	LOW E is False	DIA is False	
COL is False	LOW E is True	DIA is True	
COL is False	LOW E is True	DIA is False	
COL is False	LOW E is False	DIA is True	
COL is False	LOW E is False	DIA is False	

If OptionExists("COL") AND (OptionExists("LOW E") OR OptionExists("DIA"))

			The statement is
COL is True	LOW E is True	DIA is True	
COL is True	LOW E is True	DIA is False	
COL is True	LOW E is False	DIA is True	
COL is True	LOW E is False	DIA is False	
COL is False	LOW E is True	DIA is True	
COL is False	LOW E is True	DIA is False	
COL is False	LOW E is False	DIA is True	
COL is False	LOW E is False	DIA is False	

If (OptionExists("COL") OR OptionExists("LOW E")) OR OptionExists("DIA")

			The statement is
COL is True	LOW E is True	DIA is True	
COL is True	LOW E is True	DIA is False	
COL is True	LOW E is False	DIA is True	
COL is True	LOW E is False	DIA is False	
COL is False	LOW E is True	DIA is True	
COL is False	LOW E is True	DIA is False	
COL is False	LOW E is False	DIA is True	
COL is False	LOW E is False	DIA is False	

Scripting Expressions Examples

Option Visibility: In this example, the option will be hidden unless the attribute "LSIGAS" or "KSIGAS" returns a value of "SSB"

```
    'ENTER YOUR CODE HERE!
    retval = False
    If (Attributes("LSIGAS").value = "SSB") Or (Attributes("KSIGAS").value =
"SSB") Then
        retval = True
    End If

    'DO NOT MODIFY CODE BELOW THIS LINE!
```

Option Default Value Expression: In this example, the option code's default input is pulled from a lookup table based on the window's H and W.

```
    'ENTER YOUR CALCULATION HERE!!
    retval = TableLookup("GRIDS DH", Attributes("W").value,
Attributes("H").value)

    'DO NOT MODIFY CODE BELOW THIS LINE
```

Attribute Script: This example, returns if the glass is tempered or annealed on the Lock Sash by looking at the glass package that was selected in Options Wizard.

```
    'ENTER YOUR CALCULATION HERE!!
    'Determine if the lock sash's IG is tempered or annealed
    Dim typ1 As String

    'Fiqure out if the glass is tempered or not
    If OptionExists("XGLS") Then
        typ1 = GroupCode("GLASS PACKAGE LS")
        typ1 = Mid(typ1, 5, 1)
    Else
        typ1 = GroupCode("GLASS PACKAGE")
        typ1 = Right(typ1, 1)
    End If

    'Output the glass process
    If typ1 = "A" Then
        retval = "ANN"
    Else
        retval = "TMP"
    End If

    'DO NOT MODIFY CODE BELOW THIS LINE
```

Extended Functions

The FeneVision® scripting interface not only supports all the major conditional, mathematical and operational statements and functions supported in VBA but also has been expanded to provide access to relevant order information through the use of extended functions and properties. These functions and properties are listed under 'Extended Functions' in Appendix A.

Functions:

```
IsOptionValue

        Syntax
                retval = IsOptionValue (Code, Value, Group)
                Searches the list of ordered item options for the specified option code,
                group, and value and returns True if found otherwise returns False.

        Return Value
                Returns retval a Boolean value.

        Parameters
                Code A String that evaluates to the option code.
                Value A Variant that evaluates to the option value.
                Group A String that evaluates to the option group or question (optional).

        Example
                retval = IsOptionValue("PATTERN", "1Vx2H","MUNTIN)

OptionValue

        Syntax
                retval = OptionValue( Code, Group )
                Searches the list of ordered item options for the specified option code
                and group and returns the corresponding option value if found.

        Return Value
                Returns retval a variant value.

        Parameters
                Code A String that evaluates to the option code.
                Group Optional.  A string that evaluates to the option group or question.

        Example
                If OptionExists ("CMR") Then
                        retval = OptionValue("CMR") - 1.125
                        Else
                        retval = (retval/2) - 1.125
                End If

OptionExists

        Syntax
                retval = OptionExists( Code, Group )
                Searches the list of ordered item options for the specified option code
                and group and returns True if found otherwise returns False.
```

Return Value
Returns *retval* a Boolean value.

Parameters
Code A String that evaluates to the option code.
Group Optional. A String that evaluates to the option group or question.

Example
```
retval = OrderedHeight
if OptionExists("OS") Then
      retval = retval – 0.25
End If
```

CurrentOption

Syntax
value=CurrentOption.Value
Used to pull the current option information into a default option, default option value or visibility script.

Example
```
retval = CurrentOption.Code
```

CurrentQuestion

Syntax
value=CurrentQuestion.Value
Used to pull the current question information into a default option, default option value or visibility script.

Example
```
retval=CurrentQuestion.Question
```

GroupExists

Syntax
retval = GroupExists(*Group*)

Searches the list of ordered item groups for the specified option group and returns True if found otherwise returns False.

Return Value
Returns *retval* a Boolean value.

Parameters
Group A String that evaluates the option group.

Example
```
retval = OrderedThickness
If GroupExists("Muntin") Then
      retval = retval + 0.25
End If
```

GroupCode

Syntax

retval = GroupCode (*Group*)

Searches for the existence of an option code in the specified group of option codes, returns the first option found in the group otherwise returns an empty string.

Return Value

Returns *retval* a String value.

Parameters

Group A String that evaluates the group.

Example

```
retval = OrderedThickness
If GroupCode ("Muntins") <> "" Then
      retval = retval + .0375
End If
```

CustomValue

Syntax

retval = CustomValue (Prename, *Code*, *Group*)

Searches the list of ordered item option custom parameters for the specified parameter name, option code and group and returns the corresponding parameter value if found.

Return Value

Returns *retval* a String value.

Parameters

ParamName A String that evaluates to the parameter name.
Code Optional. A String that evaluates to the option code.
Group Optional. A String that evaluates to the option group or question.

Example

```
retval = CustomValue("lsig_hcount ")
```

CustomExists

Syntax

retval = CustomExists(ParamName, *Code*, *Group*)

Searches the list of ordered item custom options for the specified custom option and returns True if found otherwise returns False.

Return Value

Returns *retval* a Boolean value.

Parameters

ParamName A String that evaluates to the parameter name.
Code Optional. A String that evaluates to the option code.
Group Optional. A String that evaluates to the option group or question.

Example

```
If CustomExists("lsig_hcount") Then
```

```
                retval = CustomValue("lsig_hcount ")
            End If
```

OrderedWidth

Syntax

retval = OrderedWidth
Returns the ordered item width.

Return Value

Returns *retval* a Single value.

Example

```
retval = OrderedWidth
if OptionExists("OS") Then
    retval = retval - 0.25
End If
```

OrderedHeight

Syntax

retval = OrderedHeight
Returns the ordered item Height.

Return Value

Returns *retval* a Single value.

Example

```
retval = OrderedHeight
if OptionExists("OS") Then
    retval = retval - 0.25
End If
```

OrderedThickness

Syntax

retval = OrderedThickness
Returns the ordered item thickness.

Return Value

Returns *retval* a Single value.

Example

```
retval = OrderedThickness
if OptionExists("OS") Then
    retval = retval - 0.25
End If
```

OrderedCallSize

Syntax

retval = OrderedCallSize
Returns the ordered item call size.

Return Value

Returns *retval* a Single value.

Example

```
retval = OrderedCallSize
```

```
if retval = "3040" Then
      …
End If
```

OrderComment

Syntax
retval = OrderComment
Returns the order comment.

Return Value
Returns *retval* a String value.

Example
retval = OrderComment

ItemComment

Syntax
retval = ItemComment
Returns the ordered item line comment.

Return Value
Returns *retval* a String value.

Example
retval = ItemComment

Left

Syntax
retval = Left(*String*, *Length*)
Returns a string with only the *Length* of characters from the left of *String*.

Return Value
Returns *retval* a String value.

Parameters
String is the source string containing the characters originating from the left portion.
Length is the number of characters that will be returned.
If the length of *String* is less than *Length*, an error will be returned in the system.

Example
Left("APPLE", 3) returns "APP"
Left("APPLE", 4) returns "APPL"
Left("APPLE", 6) returns a scripting engine error

Right

Syntax
retval = Right(*String*, *Length*)
Returns a string with only the *Length* of characters from the right of *String*.

Return Value
Returns *retval* a String value.

Parameters

String is the source string containing the characters originating from the left portion.
Length is the number of characters that will be returned.
If the length of *String* is less than *Length*, an error will be returned in the system.

Example

Right("APPLE", 3) returns "PLE"
Right("APPLE", 4) returns "PPLE"
Right("APPLE", 6) returns a scripting engine error

Mid

Syntax

retval = Mid(*String*, Start, *[Length]*)
Returns a string with only the *Length* of characters starting with the character at position *Start*.

Return Value

Returns *retval* a String value.

Parameters

String is the source string containing the characters originating from the left portion..
Start is the position to start returning characters
Length is the number of characters to return starting from position *Start*. If omitted, the function will return to the end of the *String*.
If the length of the remaining characters in *String* beyond the Start position is less than the *Length*, an error will be returned in the system.

Example

Mid("APPLE", 2, 3) returns "PPL"
Mid("APPLE", 4) returns "LE"
Mid("APPLE", 4, 3) returns a scripting engine error

Len

Syntax

retval = Len(*String*)
Returns a string with only the *Length* of characters starting with the character at position *Start*.

Return Value

Returns an Integer value.

Parameters

String is the source string containing the characters that you pull the right portion of.
Start is the position to start returning characters
Length is the number of characters to return starting from position *Start*. If omitted, the function will return to the end of the *String*.

originating from the left portion.
Start is the position to start returning characters
Length is the number of characters to return starting from position *Start*. If omitted, the function will return to the end of the *String*. If the length of the remaining characters in *String* beyond the Start position is less than the *Length*, an error will be returned in the system.

Example
Len("APPLE", 2, 3) returns "PPL"
Len("APPLE", 4) returns "LE"
Len("APPLE", 4, 3) returns a scripting engine error

Round

Syntax
retval = Round(*Number*, *Digits*)
Rounds a number to a specified number of digits.

Return Value
Returns *retval* a Double value.

Parameters
Number is the value that needs to be rounded.
Digits are the number of digits you want the number rounded to.
If *Digits* is greater than zero, then the number is rounded up to the specified number of decimal places.
If *Digits* is zero, then the number is rounded to the nearest integer.
If *Digits* is less than zero, then the number is rounded to the left of the decimal point.

Example
Round(2.37 , 1) equals 2.4
Round(5.247 , 1) equals 5.2
Round(-1.675 , 2) equals -1.68
Round(41.5 , -1) equals 40

RoundUp

Syntax
retval = RoundUp(*Number*, *Digits*)
Rounds a number up, away from zero
If *Digits* is greater than zero, then the number is rounded up to the specified number of decimal places.
If *Digits* is zero, then the number is rounded up to the nearest integer.
If *Digits* is less than zero, then the number is rounded up to the left of the decimal point.

Return Value
Returns *retval* a Double value.

Parameters
Number is any real number that needs to be rounded up.
Digits are the number of digits you want the number rounded to.

Example
RoundUp(5.2 , 0) equals 6
RoundUp(43.9 , 0) equals 44
RoundUp(5.23159 , 3) equals 5.232

RoundUp(-5.23159 , 1) equals -5.3
RoundUp(6.1276, -2) equals 61,300

RoundDown

Syntax

retval = RoundDown(*Number*, *Digits*)
Rounds a number down, towards zero
If vnDigits is greater than zero, then the number is rounded down to the specified number of decimal places.
If vnDigits is zero, then the number is rounded down to the nearest integer.
If vnDigits is less than zero, then the number is rounded down to the left of the decimal point.

Return Value

Returns *retval* a Double value.

Parameters

Number is any real number that needs to be rounded down.
Digits are the number of digits you want the number rounded to.

Example

RoundDown(5.2 , 0) equals 5
RoundDown(43.9 , 0) equals 43
RoundDown(5.23159 , 3) equals 5.231
RoundDown(-5.23159 , 1) equals -5.2
RoundDown(6.1276, -2) equals 61,200

MRound

Syntax

retval = MRound(*Number*, *Multiple*, *RoundType*)
Rounds a number to a desired multiple
Mround, when *RoundType* = 0 or is omitted, rounds up, away from zero, if the remainder of the dividing number by multiple is greater than or equal to half of the value of the multiple. If *RoundType* = 1 then the number will always round up, away from zero. If *RoundType* = -1 then the number will always round down, toward zero.

Return Value

Returns *retval* a Double value.

Parameters

Number is the number to be rounded.
Multiple is the multiple to which you want to use to round the number.
RoundType is an optional argument, which specifies whether to always round up, always round down, or to use normal rounding practices.

Example

MRound(10 , 3) or MRound(10 , 3, 0) equals 9
MRound(-10 , -3) or MRound(-10 , -3, 0) equals -9
MRound(1.3 , 0.2) or MRound(1.3 , 0.2, 0) equals 1.4
MRound(1.3 , 0.2, -1) equals 1.2
MRound(1.2 , 0.5, 1) equals 1.5
MRound(5 , -2) equals #NUM!

If the length of the remaining characters in *String* beyond the Start position is less than the *Length*, an error will be returned in the system.

FracToDec

Syntax
retval = FracToDec(*Frac*)
Converts a fractional number to its decimal equivalent.

Return Value
Returns *retval* a Double value.

Parameters
Frac A String value that equates to the fractional value.

Example
retval = FracToDec("2 ¼") equals 2.25
retval = FracToDec("2") equals 2

DecToFrac

Syntax
retval = DecToFrac(*Dec*, *Precision*)
Converts a decimal number to its fractional equivalent.

Return Value
Returns *retval* a String value.

Parameters
Dec A Double value that equates to the decimal value.
Precision Optional. A Long value that equates to the minimum fraction to convert to, defaults to 64.

Example
retval = DecToFrac("2.25") equals 2 ¼
retval = DecToFrac("2") equals 2

CheckInventory

Syntax
retval = CheckInventory(*PartNo*, *PartNoSuffix*, *Quantity*, *Width*, *Height*)
Checks the on hand quantity of the specified part and returns True if the required quantity can be fulfilled.
Note1: The required quantity is calculated by multiplying the order line item release quantity with the Quantity specified.
Note2: Standard multiplier conventions are used: Quantity To Check = required quantity * [Width] * [Height] * Primary Stocking Multiplier

Return Value
Returns *retval* a Boolean value.

Parameters
PartNo Optional. A String that evaluates to the part number. Defaults to part number of current part if not specified.
PartNoSuffix Optional. A String that evaluates to the part number suffix. Defaults to part number suffix of current part if not specified.
Quantity Optional. A Single that evaluates to the part quantity. Defaults to quantity of current part if not specified.
Width Optional. A Single that evaluates to the part width. Defaults to width of current part if not specified.
Height Optional. A Single that evaluates to the part height. Defaults to height of current part if not specified.

Example

```
retval = CheckInventory()
retval = CheckInventory("SSCL20x30")
retval = CheckInventory("SSCL20x30", "0000", 2)
```

AllocateInventory

Syntax

retval = AllocateInventory(*PartNo*, *PartNoSuffix*, *Quantity*, *Width*, *Height*)
Removes the required quantity of the specified part from the on hand quantity of the specified part and returns True if the allocation is successful.

Note1: The required quantity is calculated by multiplying the order line item release quantity with the Quantity specified.
Note2: Standard multiplier conventions are used: Quantity To Check = required quantity * [Width] * [Height] * Primary Stocking Multiplier
Note3: The on hand quantity of the part is determined at the beginning of the schedule release.
Note4: The adjusted part on hand quantity is not saved to the live inventory levels of the part. The live inventory levels of the part are still maintained by the Allocation and Relieve schedule maintenance functions.
Note5: The parts being allocated by this function need to exist in the bill of materials of the part being released.

Return Value

Returns *retval* a Boolean value.

Parameters

PartNo Optional. A String that evaluates to the part number. Defaults to part number of current part if not specified.
PartNoSuffix Optional. A String that evaluates to the part number suffix. Defaults to part number suffix of current part if not specified.
Quantity Optional. A Single that evaluates to the part quantity. Defaults to quantity of current part if not specified.
Width Optional. A Single that evaluates to the part width. Defaults to width of current part if not specified.
Height Optional. A Single that evaluates to the part height. Defaults to height of current part if not specified.

Example

```
retval = AllocateInventory()
retval = AllocateInventory ("SSCL20x30")
retval = AllocateInventory ("SSCL20x30", "0000", 2)
```

IsSubLineItem

Syntax

retval = IsSubLineItem
Returns True if the ordered item is a sub line item of another ordered item, otherwise returns False.

Return Value

Returns *retval* a Boolean value.

Example

```
If IsSubLineItem = True Then
      retval = MainLineItem.OrderedWidth
```

```
            End If
```

TableLookup

Syntax

retval = TableLookup(TableName, x, *[y]*)
Returns a string value that represents a cell in a Lookup Table.

Return Value

Returns *retval* a String value.

Parameters

TableName is the name of the Lookup Table to query.
x is the column header to look for when querying the data from the Lookup Table
y is the row header to look for when querying the data from the Lookup Table. If omitted, the function will look for a row header containing empty string (""). For 1D Lookup Tables, the *y* parameter should be omitted. For 2D Lookup Tables, the query will not find a matching result, and return ""

If the query can't find matching x and/or y column headers, the function returns empty string ""

Example

Table NFRC–DH

	CPD	RUF	RSHG
CLCL–DSB–NG	ABC–K–2–2222–0000	.39	.40
CLCL–DSB–GR	ABC–K–2–2222–0001	.39	.39
CLLE–DSB–GR	ABC–K–2–2245–0000	.32	.30

TableLookup("NFRC–DH", "CPD", "CLCL–DSB–NG") returns "ABC–K–2–2222–0000"
TableLookup("NFRC–DH", "RUF", "CLCL–DSB–NG") returns ".39"
TableLookup("NFRC–DH", "CPD") returns ""

Properties:

Include

Syntax

Include = {True | False}
Used to set whether the current part is included in the material list for the ordered item.

Example

```
Include = OptionExists("W") Or OptionExists("T") Or OptionExists("WE")
```

Parent

Syntax

For write access:
Parent.object = value

```
            For read only access:
            value = Parent.object
            Used to gain reference to the current parts 'Parent' part.

      Example
            retval  = Parent.Attributes("W").Value - 3.125
```

Attributes

```
      Syntax
            For write access:
            object.Attributes(item).Value = value
            For read only access:
            value  = object.Attributes(item).Value
            Used to gain reference to the current or 'Parent' parts attributes
            collection and return or set an attribute value.

      Return Value
            Returns value a Variant value.

      Parameters
            item A String that evaluates to the Attribute name.
            object Optional.  If omitted takes on the current part otherwise the
            'Parent' property can be specified.

      Example
            AREA = (Attributes("IGH").Value / 12) * (Attributes("IGW").Value / 12)
            or
            Include = Parent.Attributes("W").Value > 25.875
```

CurrentPart

```
      Syntax
            For read only access:
            value = CurrentPart.Value
            Used to pull the current part information into an attribute.

      Example
            retval = CurrentPart.PartNo
```

MainLineItem

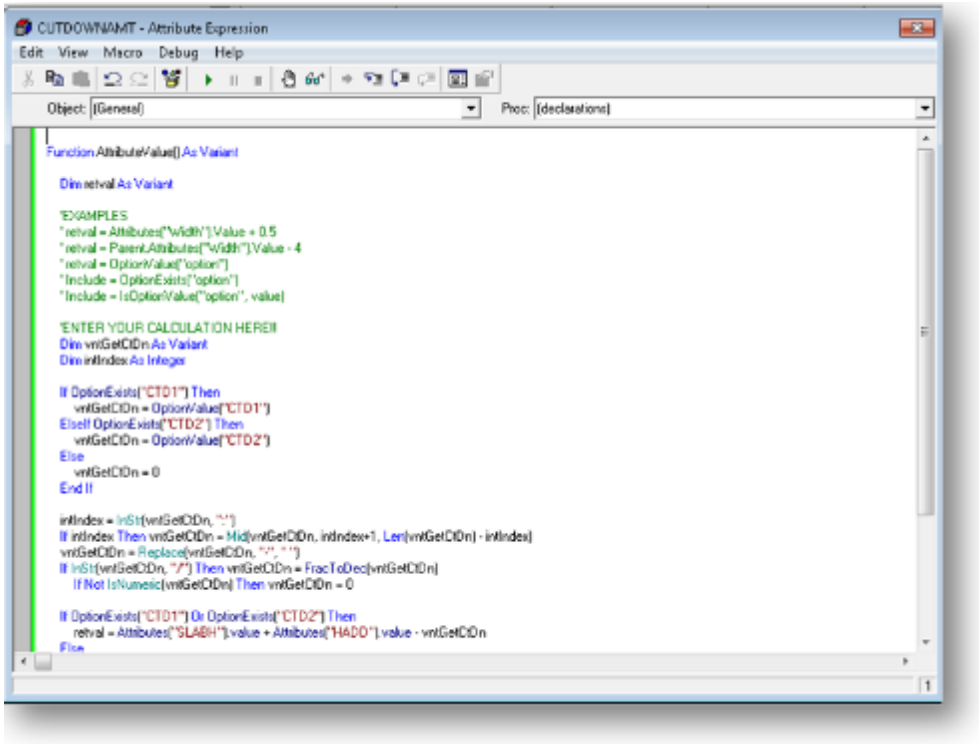
```
      Syntax
            For read only access:
            value = MainLineItem.Attributes(item).Value
            Used to gain reference to the main line item ordered part, if the current
            ordered item is a sub line item.  All existing scripting functions and
            properties can be accessed on the main line item.

      Examples
            retval = MainLineItem.OrderedWidth
            retval = MainLineItem.OrderedHeight
            Include = MainLineItem.OptionExists("WHT")
            retval = MainLineItem.Attributes("W").Value
```

Order

```
      Syntax
            For read only access:
```

```
        value = Order.[Property Name]
        Used to gain reference to the order header, if it exists.  Various
        properties on the order header can then be accessed.

Return Value
        Returns an object reference.

Properties
        Comments
        CustomerID
        CustomerRef
        OrderDate
        MfgCustomerID
        PONumber
        ReqDate
        SiteID

Example
        'Returns the customer ID.
        retval = Order.CustomerID

Comments
```

The Order object is only available during production schedule releases. It is not available during part breakdowns. Note that all of the properties will contain empty values if the Order object does not exist. It is a good practice to first check if the Order object is "Nothing" before accessing its properties. MfgCustomerID allows users to specify a customer ID that applies to both FeneVision® CORE and FeneVision® WEB.



Truth Table Answer Key

If (OptionExists(“COL”) AND OptionExists(“LOW E”)) OR OptionExists(“DIA”)

			The statement is
COL is True	LOW E is True	DIA is True	True
COL is True	LOW E is True	DIA is False	True
COL is True	LOW E is False	DIA is True	True
COL is True	LOW E is False	DIA is False	False
COL is False	LOW E is True	DIA is True	True
COL is False	LOW E is True	DIA is False	False
COL is False	LOW E is False	DIA is True	True
COL is False	LOW E is False	DIA is False	False

If OptionExists(“COL”) AND (OptionExists(“LOW E”) OR OptionExists(“DIA”))

			The statement is
COL is True	LOW E is True	DIA is True	True
COL is True	LOW E is True	DIA is False	True
COL is True	LOW E is False	DIA is True	True
COL is True	LOW E is False	DIA is False	False
COL is False	LOW E is True	DIA is True	False
COL is False	LOW E is True	DIA is False	False
COL is False	LOW E is False	DIA is True	False
COL is False	LOW E is False	DIA is False	False

If (OptionExists(“COL”) OR OptionExists(“LOW E”)) OR OptionExists(“DIA”)

			The statement is
COL is True	LOW E is True	DIA is True	True
COL is True	LOW E is True	DIA is False	True
COL is True	LOW E is False	DIA is True	True
COL is True	LOW E is False	DIA is False	True
COL is False	LOW E is True	DIA is True	True
COL is False	LOW E is True	DIA is False	True
COL is False	LOW E is False	DIA is True	True
COL is False	LOW E is False	DIA is False	False