### Runtime Options
- **Must be specified for each module**, before any executable code.
  - Option Explicit
    - Imposes explicit variable declaration.
  - Option BASIC.behaves like VBA.
  - Option VBA Support
    - Activates VBA support.
  - Option Base
    - Arrays are 1-indexed instead of 0-indexed.
  - Option Class Module
    - To use for classes creation (+ Option Compatible).

### BASIC Constants
- **True** = (Boolean) = 1
- **False** = (Boolean) = 0
- **Pi** = 3.14159265358979
- **Null** = Nothing
- **Uintialized** = any value.
- **Empty** = any object (objects) suppresses any previous assignment.

### Functions
- Functions syntax: **Result** = **FuncName**(arguments)
- **String Functions (type String)**
  - **Asc()**
    - Returns the ASCII value of the 1st character of a string.
    - Asc("Azerty") = 65
  - **Chr()**
    - Returns the character which ASCII code is passed.
    - Chr(65) = "A"
  - **Left()**
    - Returns a string position within another. If not found, returns 8.
    - Left("LibreOffice", "Office") = 6
  - **Right()**
    - Returns a string from an array of strings.
    - Right(MyArray, "") = "C:\Dir\SubDir\MyFile.ods"

### Numerical Functions
- **Abs()**
  - Returns a number absolute value.
  - Abs(-65) = 65
- **Exp()**
  - Returns e to a given power.
  - Exp(0) = 1
  - Exp(2) = 7.38905609896064
- **Fix()**
  - Returns an integer part of a number (no rounding).
  - Fix(3.14159265) = 3
- **Hex()**
  - Returns the hexadecimal value of a decimal number.
  - Hex(10) = "A"
- **Int()**
  - Returns a number integer logarithm.
  - Int(3.14159265) = 3
- **Log()**
  - Returns the natural logarithm.
  - Log(10) = 2.30258509299404
- **Oct()**
  - Returns the octal value of a decimal number.
  - Oct(10) = "12"

### Randomize()
- Initializes the random number generator (before using Rand()).
  - Randomize(45)

### Trigonometrical Functions
- **Cos()**
  - Calculates the cosine of an angle.
  - Cos(90°) = 0
- **Sin()**
  - Calculates the sine of an angle.
  - Sin(90°) = 1
- **Tan()**
  - Calculates a number square root.
  - Tan(45°) = 1

### Date/Time Functions
- **NOW()**
  - Returns the current date and time (Date type).
  - Now() = Today
- **DateAdd()**
  - Returns a new date from a starting date and an addition (date arithmetic). If not found, returns 8.
  - DateAdd("m", 1, Now() = 8/14/2017)
- **Date()**
  - Returns the current date (Date type).
  - Date() = Today
- **DateDiff()**
  - Calculates a dates difference, expressed in the desired unit.
  - DateDiff("d", "8/14/2017", "7/14/2017") = 1
- **DateValue()**
  - Returns a date numerical value, calculated from its 3 parts (year, month and day).
  - DateValue("7/14/2017") = 2017-07-14
- **DateSerial()**
  - Returns a date value from its string representation.
  - DateSerial("7/14/2017") = 2017-07-14
- **DatePart()**
  - Returns a date part from a date structure.
  - DatePart("d", "7/14/2017") = 14
- **DateSerial()**
  - Returns a date as a UNO Date type value.
  - DateSerial("7/14/2017") = 2017-07-14
- **CDate()**
  - Converts a UNO Date type value into a Date type value.
  - CDate("7/14/2017") = 2017-07-14
- **CDateFromISO()**
  - Converts a file name in OS form to URL form.
  - ConvertFromISO("C:\Dir\SubDir\MyFile.ods") = file:///c:/somedir/file.ods
  - CDateFromISO("20170714") = 2017-07-14
- **CDateFromISO()**
  - Converts a UNO Date type value into a Date type value.
  - CDate("7/14/2017") = 2017-07-14
- **CDateToISO()**
  - Converts a file name in URL form to OS form.
  - ConvertToISO("file:///c:/somedir/file.ods") = C:\Dir\SubDir\MyFile.ods
  - CDateToISO("file:///c:/somedir/file.ods") = C:\Dir\SubDir\MyFile.ods
Colors are stored as Longs. 
Red(), Green(), Blue() Extracts the said colour component. 
RGB() Returns a color from its 3 components red, green and blue. 
RGB(128, 0, 0) → #800000 (red)

Array Functions
-Array() Creates an array from discrete values. 
MyArray = Array("One", 2, Now())
-DimArray() LikeArray(): MyArray = DimArray("One", 2, Now())
-Use only if implicit variable declaration, otherwise use Array().
-Erase (Instruction) Erases an array contents. In case of a dynamic array, frees the memory. Erase MyArray
-LBound() Lower bound. 
UBound() Upper bound.

Type Information Functions
These functions give information about the variables.

Any Variable
- TypeName() Returns a string that details a given variable.
- VarType() Returns a numerical identifier for a given variable.
- IsNull() Returns True if the argument is a UNO structure.
- IsNull(Struct) Returns True if the argument is a UNO structure.

The first two functions return one of the values below:

VarType | TypeName | VarType | VarType |
---|---|---|---|
0 | Empty | 5 | Double |
1 | Null | 6 | Currency |
2 | Integer | 7 | Date |
3 | Long | 8 | String |
4 | Single | 9 | Object |

Arrays: bid2 vartype

Variants
- Return True according to the actual type found.
- Function | Type check | Function | Type check |
---|---|---|---|
CArray | Array | CIsnull | Null (no data). |
CDate | Date | CIsNumeric | Numerical value. |
CEmpty | Uninitialized variable. | CIsObject | OLE object. |
CError | Error value | CIsUNOStruct | True if UNO structure. |

UNO Structures And Objects
- CreateUnoServiceName() Creates a UNO service. *Name* is case-sensitive!
- IsUNOStruct() True if UNO structure. 
- (struct.) Obj Properties Returns the UNO structure name (String).
- HasUnoInterfaces() True if UNO object supports interfaces.
- (obj.) SupportsService() True if (UNO obj) supports the service in argument (String).
- EqualUnoObjects(a1, a2) True if both var. refer to the same object instance.

Typecast Functions
These functions convert a value from a compatible type into another. The function name reflects the target type name.

- CBoolean() To Boolean
- CByte() To Byte
- CCur() To Currency
- CDate() To Date
- CHalf() To Half
- CInt() To Integer
- CVar() To Variant
- CDouble() To Double
- CStr() To String
- CRound() To Round
- CShort() To Short
- CSingle() To Single
- CStr() To String
- CSigned() To Signed
- CUnsigned() To Unsigned
- CString() To String
- CInt64() To Int64

- Error Information Functions
Erl | Error line number. | Error | Error message. |
- ErrorCode() Error code.

Misc. Functions
- GetGUIType() Returns a value that reflects the OS, among: 
- Command | Style | Param, Synchro |
---|---|---|
Shell | (Comme, Style, Param, Synchro) | Command |
With | Command the command to execute (String). |
Style | The window in which the process takes place, among (Integer): |
- Command Execution parameters to hand to the command (String). |
- Param Execution flag: True | Wait for the command execution to finish. |
| False | Do not wait for the command execution to finish. |

Format Function – Formatting Masks
The Format() function converts a number into a string by formatting it according to a mask.

- A format mask is a string that can be split in 3 sections separated with semicolons: 
  \text{val}:@\text{val}:@\text{val}:@. One section only = all numbers.

- \text{Language formatting of numbers: Tools Options Language settings Languages}.

| B | 8 | Number is mandatory at that position (0 if missing) |
| E | 8 | Optional number |
| + | 8 | Decimal separator (6.4) |
| - | 8 | Literal character, appears as-is in the result. |


\text{Scientific format.}

\text{Escape character: the character that follows is in the result as-is.}

\text{Result in percent format.}

\text{Quarter number (1 or 2 char)}

\text{Week number (1 or 2 char)}

\text{Minute number (1 or 2 char)}

\text{Seconds (1 or 2 char)}

\text{More details in the on-line help.}

\text{End Enum}

\text{Name} is case-sensitive!

\text{Enumeration names and value names must be unique within a library and across modules.}

\text{VBA Support does not complete.}

\text{Save original Basic code}

\text{VBA Functions}

\text{AsCm} | FV() | IRR() | Round() |
| Chrm() | Input() | Me() | RTL() |
| DDB() | InStrRev() | MIRR() | StrReverse() |
| FormatDateTime() | IFmt() | NPer() | WeekDayName() |

\text{Details more in the on-line help.}

\text{Enumeration values are rendered as Long.}

\text{Name enumeration values and names names must be unique within a library and across modules.}

\text{Credits}

\text{Author: Jean-François Nifenecker – jean-francois.nifenecker@laposte.net}

\text{We are like dwarves perched on the shoulders of giants, and thus we are able to see more and farther than the latter. And this is not at all because of the acuteness of our sight or the stature of our body, but because we are carried aloft and elevated by the magnitude of the giants. (Bernard of Chartres [attr.])}

\text{History}

\text{Version | Date | Comments}
| 1.10 | 04/16/2018 | First EN version |
| 1.14 | 26/04/19 | Updates and fixes. |

\text{License}

\text{This ReMix Code is placed under the CreativeCommons BY-SA v3 (fr) license Information}

\text{https://creativecommons.org/licenses/by-sa/3.0/fr/}