LibreOffice RefCard

LibreOffice BASIC
Runtime Library

v. 1.14 – 04/26/2019

Beginner

Written with LibreOffice v. 5.3.3 – Platform : All

Runtime Options

Must be specified **for each module**, before any executable code.

`Option Explicit`	Imposes explicit variable declaration.
`Option Compatible`	LibO BASIC behaves like VBA.
`Option VBASupport 1`	Activates VBA support.
`Option Base 1`	Arrays are 1-indexed instead of 0-indexed.
`Option ClassModule`	To use for classes creation (+ `Option Compatible`).

BASIC Constants

`True`	True (Boolean)
`False`	False (Boolean)
`Pi`	`3.14159265358979` (Double)
`Empty`	Unitialized value.
`Null`	The var. doesn't hold any useful data.
`Nothing`	(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` See `Chr()`, ASCII table.
`Chr()`	Returns the character which ASCII code is passed. `Chr(65)` → `"A"` See `Asc()`, ASCII table.
`ConvertFromURL()`	Converts a file name in URL form to OS form. URL form: `protocol:///host/path/to/file.ext` Ex. Windows : `file:///c:/somedir/file.ods` Ex. Linux : `file:///home/user/somedir/file.ods`
`ConvertToURL()`	Converts a file name in OS form to URL form. See `ConvertFromURL()`
`Format()`	Converts a number into a string, with mask formatting. On the 7/14/2017, `Format(Now(), "yyyy")` → `"2017"` See Format function – Formatting Masks.
`InStr()`	Returns a string position within another. If not found, returns `0`. `InStr("LibreOffice", "Office")` → `6`

`Join()`	Returns a string from an array of strings.
	`MyArray = Array("C:", "Dir", "SubDir", "MyFile.ods")`
	`Join(MyArray, "\")` → `"C:\Dir\SubDir\MyFile.ods"`
	See `Split()`

`LCase()`	Returns a string in lower case. `LCase("LibreOffice")` → `"libreoffice"` See `UCase()`
`Left()`	`Left(chaine, N)` Extracts `N` characters from the left of a string. `Left("LibreOffice", 5)` → `"Libre"` See `Mid()`, `Right()`
`Len()`	Returns the number of characters in a string. `Len("LibreOffice")` → `11`
`LTrim()`	Suppresses the leftmost spaces from a string. See `RTrim()`, `Trim()`
`Mid()`	`Mid(chaine, P, N)`. Extracts `N` characters in a string, starting at position `P`. `Mid("14/7/2017", 4, 1)` → `"7"` See `Left()`, `Right()`
`Right()`	`Right(chaine, N)`. Extracts `N` characters from the right of a string. See `Left()`, `Mid()`
`RTrim()`	Suppresses the rightmost spaces from a string. See `LTrim()`, `Trim()`
`Space()`	Returns a string made of a series of spaces. `Space(3)` → `"   "` See `String()`

`Split()`	Returns an array of strings from a single string, separating at a given character.
	`MyString = "C:\Dir\SubDir\MyFile.ods"`
	`Split(MyString, "\")` → a 4 items array: `"C:"`, `"Dir"`, `"SubDir"`, `"MyFile.ods"`
	See `Join()`

`Str()`	Converts a numeric expression into a string. `Str(-65)` → `" -65"` ☞ A space is at the left of the text. Decimal separator is a dot. See `CStr()`, `Val()`
`StrComp()`	Compares two strings and returns an integer value that represents the comparison result.
`String()`	Creates a string made of N times a character. `String(4, "Y")` → `"YYYY"` See `Space()`
`Trim()`	Suppresses the leftmost and rightmost spaces from a string. See `LTrim()`, `RTrim()`
`UCase()`	Returns a string in upper case. `UCase("LibreOffice")` → `"LIBREOFFICE"` See `LCase()`
`Val()`	Converts a string into a numerical value (`0` when not possible). `Val("12,34")` → `12,34` See `Str()`, `Val()`

Numerical Functions

`Abs()`	Returns a number absolute value.
`Exp()`	Exponential. Returns e to a given power.
`Fix()`	Returns the integer part of a number (no rounding).
`Hex()`	Returns the hexadecimal value of a decimal number.
`Int()`	Returns a number integer part (rounded to the lower value).
`Log()`	Returns a number logarithm.
`Oct()`	Returns the octal value of a decimal number.
`Randomize()`	Initializes the random number generator (before using `Rnd()`).
`Rnd()`	Returns a random number, between `0` and `1`. See `Randomize()`
`Sgn()`	Returns the sign of a number.
`Sqr()`	Calculates a number square root.

Trigonometrical Functions

Angles in radians. `radians = (degrees * Pi)/180`

`Atn()` Arc tangent. `Cos()` Cosine. `Tan()` Tangent.

Date/Time Functions

"UNO" Date Functions

LibreOffice API often uses "UNO" dates, that is of type `com.sun.star.util.DateTime` (or `.Date` or `.Time`), structured as follows:

`IsUTC`	`True` if timezone is UTC.	`Hours`	Hours (0-23).
`Year`	Year number	`Minutes`	Minutes (0-59).
`Month`	Month number (`0` if empty).	`Seconds`	Seconds (0-59).
`Day`	Day number (`0` if empty).	`NanoSeconds`	Nanoseconds.

☞ `Date` ↔ Uno Date : use the conversion functions `CDateXxx` below.

Date/Time Functions

`CDateFromISO()`	Returns the Date type value corresponding to the date ISO string (`YYYYMMDD`). `CDateFromISO("20170714")` → that date in Date format.
`CDateFromUnoDate()`	Converts a UNO `com.sun.star.util.Date` structure into a Date type value.
`CDateFromUnoDateTime()`	Converts a UNO `com.sun.star.util.DateTime` structure into a Date type value.
`CDateFromUnoTime()`	Converts a UNO `com.sun.star.util.Time` structure into a Date type value.
`CDateToISO()`	Returns an ISO date string (`YYYYMMDD`) from a Date type value. On 7/14/2017, `CDateToISO(Now())` → `"20170714"`
`CDateToUnoDate()`	Returns a date as a UNO `com.sun.star.util.Date` structure.
`CDateToUnoDateTime()`	Returns a date as a UNO `com.sun.star.util.DateTime` structure.
`CDateToUnoTime()`	Returns a date as a UNO `com.sun.star.util.Time` structure.
`Date()`	Returnes the current date (Date type). See `Now()`, `Time()`
`DateAdd()`	Returns a new date from a starting date and an addition criterion (±). On 7/14/2017, `DateAdd("m", 1, Now())` → `8/14/2017` Add type masks: `yyyy` Year; `q` Quarter; `m` Month; `y` Year day; `w` Week day; `ww` Week; `d` Day; `h` Hour; `n` Minute; `s` Second
`DateDiff()`	Calculates a dates difference, expressed in the desired unit (See table in `DateAdd()`). `DateDiff("m", "8/14/2017", "7/14/2017")` → `1`
`DatePart()`	Returns the specified date part (See table in `DateAdd()`). `DatePart("q", "7/14/2017")` → `3`
`DateSerial()`	Returns a date numerical value, calculated from its 3 parts year, month and day. `DateSerial(2017,7,14)` →
`DateValue()`	Returns a date value from its string representation. `DateValue("7/14/2017")` → `07/14/2017` (Date type)
`Day()`	Returns the day number in the month. `Day("7/14/2017")` → `14`
`Hour()`	Returns the current time. It is noon. `Hour(Now())` → `12`
`Minute()`	Returns the minutes of a Date type value. It is noon. `Minute(Now())` → `0`
`Month()`	Returns the month number. `Month("7/14/2017")` → `7`
`Now()`	Returns the current date and time (Date type). See `Date()`, `Time()`
`Second()`	Returns the seconds of a Date type value. It is noon. `Second(Now())` → `0`
`Time()`	Returns the current time as a Date type value. See `Date()`, `Now()`
`Timer()`	Returns a Double value with the number of elapsed seconds from midnight. ☞ Set `Timer()` to a variable before use!
`TimeSerial()`	Returns a Date type value, calculated from the 3 items hours, minutes and seconds. `TimeSerial(12,25,14)` → `12:25:14` (type Date)
`TimeValue()`	Returns an hour value (Date type) from a string value. `TimeValue("12:25:14")` → `12:25:14` (type Date)
`Wait`	(instruction) Waits the number of specified milliseconds. `Wait 1000` → pauses for 1 sec.
`WeekDay()`	Returns the week number (`1` = sunday). `Weekday("7/14/2017")` → `6` (friday)
`Year()`	Returns the year number. `Year("7/14/2017")` → `2017`

Color Functions

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)` → `8388608` (red)

Array Functions

`Array()` Creates an array from discrete values.
`MyArray = Array("One", 2, Now())`

`DimArray()` Like `Array()` : `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.
`IsUnoStruct()` Returns `True` if the argument is a UNO structure.

The first two functions return of one of the values below:

VarType	TypeName	VarType	TypeName	VarType	TypeName
0	Empty	5	Double	11	Boolean
1	Null	6	Currency	12	Variant
2	Integer	7	Date	17	Byte
3	Long	8	String	37	Decimal
4	Single	9	Object		

☞ Arrays: `8192 + vartype`

Variants

Return `True` according to the actual type found.

Function	Type check	Function	Type check
`IsArray()`	Array.	`IsNull()`	`Null` (no data).
`IsDate()`	Date.	`IsNumeric()`	Numerical value.
`IsEmpty()`	Uninitialized variable.	`IsObject()`	OLE object.
`IsError()`	Error value.	`IsUNOStruct()`	`True` if UNO structure.

UNO Structures And Objects

`CreateUnoService(Name)` Creates a UNO service. `Name` is case-sensitive!
`IsUNOStruct()` `True` if UNO *structure*.
`(struct.)Dbg_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(o1, o2)` `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.

☞ Code readability: always prefer an explicit typecast to an implicit one!

`CBool()`	To `Boolean`	`CDbl()`	To `Double`	`CSng()`	To `Single`
`CByte()`	To `Byte`	`CDec()`	To `Decimal`	`CStr()`	To `String`
`CCur()`	To `Currency`	`CInt()`	To `Integer`	`CVar()`	To `Variant`
`CDate()`	To `Date`	`CLng()`	To `Long`	`CVErr()`	To `Variant (Error)`

Error Information Functions

`Erl` Error line number. `Error` Error message.
`Err` Error code.

Misc. Functions

`GetGUIType()` Returns a value that reflects the OS, among:
1 Windows 4 OSX or Linux
3 MacOS

`GetSolarVersion()` Returns LibreOffice version.
`IsMissing()` Checks whether an optional parameter is omitted.

Calling System Commands

Command syntax: `Shell(Commande, Style, Param, Synchro)`
with:

`Command` The command to execute (`String`).
`Style` The window in which the process takes place, among (`Integer`) :
- `0` The program has focus, its window is hidden.
- `1` The program has focus and runs in a standard window.
- `2` The program has focus and runs as minimized.
- `3` The program has focus and runs as maximized.
- `4` The program starts in a standard non-focused window.
- `6` The program starts in a minimized window; focus is on the current window.
- `10` The program starts in full-screen mode.

`Param` Execution parameters to hand to the command (`String`).
`Synchro` 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: `val>0;val<0;val=0`. One section only = all numbers.

☞ Language formatting of numbers: **Tools > Options > Language settings > Languages**.

Numbers

`0`	Number is mandatory at that position (0 if missing)	`%`	Result in percent format.
`#`	Optional number	`E- E+ e- e+`	Scientific format.
`.`	Decimal separator		
`+- space ( )`	Literal character, appears as-is in the result.	`\`	Escape character: the character that follows is in the result as-is.

Dates

`D` or `DD`	Day number (1 or 2 char)	`Q` or `QQ`	Quarter number (1 or 2 char)
`M` or `MM`	Month number (1 or 2 char)	`W` or `WW`	Week number (1 or 2 char).
`MMMM`	Month name.	`h` or `hh`	Hour (1 or 2 char)
`YY` or `YYYY`	Year number (2 or 4 char)	`m` or `mm`	Minutes (1 or 2 char)
`NNN`	Day name.	`s` or `ss`	Seconds (1 or 2 char)

VBA Support

☞ VBA support is not complete.

Environment Functions

Tools > Options > LibreOffice > Load/Save > VBA Properties

Load Basic code Loads and saves VBA code from a MSOffice document into a special LibreOffice Basic module.
Executable code The VBA code is loaded, ready to execute.
Save original Basic code The document VBA code is saved apart when the document is loaded in LibreOffice.

Runtime Functions

VBA support requires the options : `Option VBASupport 1` et `Option Compatible`.

VBA Functions

`AscW`	`FV()`	`IRR()`	`Round()`
`ChrW`	`Input()`	`Me()`	`RTL()`
`DDB()`	`InStrRev()`	`MIRR()`	`StrReverse()`
`FormatDateTime()`	`IPmt()`	`NPer()`	`WeekDayName()`

More details in the on-line help.

VBA Instructions

```
Enum        Enum EnumName
              W1ND0WS   = 1 ' Windows
              OS2PM     = 2 ' OS/2 Presentation Manager
              MACINTOSH = 3 ' Macintosh
              MOTIF     = 4 ' Motif Window Manager / Unix-like
              OPENLOOK  = 5 ' Open Look / Unix-like
            End Enum
```

Enumerated values are rendered as `Long`.

Enumeration names and value names must be **unique** within a library and across modules.

ASCII Table

Dec	Hex	Val	Dec	Hex	Val	Dec	Hex	Val	Dec	Hex	Val
0	0	NUL	32	20	SPC	64	40	@	96	60	`
1	1	SOH	33	21	!	65	41	A	97	61	a
2	2	STX	34	22	"	66	42	B	98	62	b
3	3	ETX	35	23	#	67	43	C	99	63	c
4	4	EOT	36	24	$	68	44	D	100	64	d
5	5	ENQ	37	25	%	69	45	E	101	65	e
6	6	ACK	38	26	&	70	46	F	102	66	f
7	7	BEL	39	27	'	71	47	G	103	67	g
8	8	BS	40	28	(	72	48	H	104	68	h
9	9	HT	41	29	)	73	49	I	105	69	i
10	0A	LF	42	2A	*	74	4A	J	106	6A	j
11	0B	VT	43	2B	+	75	4B	K	107	6B	k
12	0C	FF	44	2C	,	76	4C	L	108	6C	l
13	0D	CR	45	2D	-	77	4D	M	109	6D	m
14	0E	SO	46	2E	.	78	4E	N	110	6E	n
15	0F	SI	47	2F	/	79	4F	O	111	6F	o
16	10	DLE	48	30	0	80	50	P	112	70	p
17	11	DC1	49	31	1	81	51	Q	113	71	q
18	12	DC2	50	32	2	82	52	R	114	72	r
19	13	DC3	51	33	3	83	53	S	115	73	s
20	14	DC4	52	34	4	84	54	T	116	74	t
21	15	NAK	53	35	5	85	55	U	117	75	u
22	16	SYN	54	36	6	86	56	V	118	76	v
23	17	ETB	55	37	7	87	57	W	119	77	w
24	18	CAN	56	38	8	88	58	X	120	78	x
25	19	EM	57	39	9	89	59	Y	121	79	y
26	1A	SUB	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	59	3B	;	91	5B	[	123	7B	{
28	1C	FS	60	3C	<	92	5C	\	124	7C	|
29	1D	GS	61	3D	=	93	5D	]	125	7D	}
30	1E	RS	62	3E	>	94	5E	^	126	7E	~
31	1F	US	63	3F	?	95	5F	_	127	7F	DEL

Credits

Author : Jean-François Nifenecker – jean-francois.nifenecker@laposte.net

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.])

History

Version	Date	Comments
1.10	04/16/2018	First EN version
1.14	26/04/19	Updates and fixes.