LibreOffice Reference Card

LibreOffice Basic
Runtime Library

v. 1.01 – October 1, 2017

Beginner

Written with LibreOffice v. 5.3.3 – Platform : All

Runtime options

Specify **per module**, before any executable code

`Option Explicit`	Force explicit declaration of variables and constants
`Option Compatible`	Makes LibreOffice Basic behave like VBA.
`Option VBASupport 1`	Enable VBA support.
`Option Base 1`	Tables and arrays are indexed from `1` instead of `0`.
`Option ClassModule`	Use to create classes (+ `Option Compatible`).

Basic constants

`True`	True (Boolean)
`False`	False (Boolean)
`Pi`	`3.14159265358979` (Double)
`Empty`	Var. non initialized.
`Null`	Var. does not have data.
`Nothing`	(objects) delete previous assignment.

Functions

☞ General function syntax: `Result = FunctionName(arguments)'`

String functions (Type String)

`Asc()`	Return the ASCII value (of 1st character) of a string. `Asc("A")` → `65` See `Chr()`, ASCII table.
`Chr()`	Return a character whose ASCII code is given. `Chr(65)` → `A` See `Asc()`, ASCII table.
`ConvertFromURL()`	Convert a file name in URL format into OS format. URL Format: `protocol:///host/path/to/file.ext` Ex. Windows : `file:///c:/folder/file.ods` Ex. Linux : `file:///home/user/folder/file.ods`
`ConvertToURL()`	Convert a file name in OS format to URI format. See `ConvertFromURL()`
`Format()`	Convert a number into a string, formatting with a mask. 7/14/2017, `Format(Now(), "yyyy")` → `"2017"` See Format function - Formatting masks.
`InStr()`	Return the position of a string inside another. If not found, return `0`. `Instr("LibreOffice", "Office")` → `6`

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

`LCase()`	Return string in lowercase `LCase("LibreOffice")` → `"libreoffice"` See `UCase()`
`Left()`	`Left(string, N)` Extracts `N` char. of a string from the left. `Left("LibreOffice", 5)` → `"Libre"` See `Mid()`, `Right()`
`Len()`	• `String` : return the number of characters of the string. `Len("LibreOffice")` → `11`
`LTrim()`	Deletes leftmost space characters of string. See `RTrim()`, `Trim()`
`Mid()`	`Mid(string, P, N)`. Extract `N` char. inside a string from position `P`. `Mid("14/7/2017", 4, 1)` → `"7"` see `Left()`, `Right()`
`Right()`	`Right(string, N)`. Extract `N` char. of a string from the right. See `Left()`, `Mid()`
`RTrim()`	Deletes rightmost space characters of string. See `LTrim()`, `Trim()`
`Space()`	Return a string consisting of several spaces. `Space(3)` → `"  "` See `String()`

`Split()`	Return a string array from a string splitting in a specific character.
	`MyString = "C:\Rep\SsRep\MyFile.ods"`
	`Split(MyString, "\")` → an array of 4 elements: `"C:"`, `"Rep"`, `"SsRep"`, `"MyFile.ods"`
	See `Join()`

`Str()`	Convert a numeric expression into a string. `Str(-65)` → `" -65"` ☞ A space precedes the text. The decimal separator is the period. See `CStr()`, `Val()`
`StrComp()`	Compare 2 strings and return an integer value representing the comparison result.
`String()`	Creates a sring with N times a character. `String(4, "Y")` → `"YYYY"` See `Space()`
`Trim()`	Remove left and right spaces from string. See `LTrim()`, `RTrim()`
`UCase()`	Return string in uppercase. `UCase("LibreOffice")` → `"LIBREOFFICE"` See `LCase()`
`Val()`	Convert a string expression into a num. value (`0` if not convertible). `Val("12.34")` → `12.34` See `Str()`

Numeric functions

`Abs()`	Return the absolute value of a number.
`Exp()`	Exponential. Return e (= 2.718...) raised to a power.
`Fix()`	Return the integer part of a number (no rounding).
`Hex()`	Return hexadecimal value of a decimal number.
`Int()`	Return the integer part of a number (rounded to the lower value).
`Log()`	Return the logarithm of a number.
`Oct()`	Return octal value of a decimal number.
`Randomize()`	Initialize the random number generator (for the `Rnd()` function).
`Rnd()`	Return a random number between `0` and `1`. See `Randomize()`
`Sgn()`	Return the sign of a number.
`Sqr()`	Return the squared root of a number.

Trigonometric functions

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

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

Date/Time functions

"UNO" date format

LibreOffice API uses "uno" dates, i.e. of type `com.sun.star.util.DateTime` (or `.Date` or `.Time`), with structure as:

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

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

Date/Time functions

`CDateFromISO()`	Return the value of type Date corresponding to date string in ISO format (`YYYYMMDD`). `CDateFromISO("20170714")` → this date in Date format.
`CDateFromUnoDate()`	Convert a UNO structure `com.sun.star.util.Date` in a value with Date type.
`CDateFromUnoDateTime()`	Convert a UNO structure `com.sun.star.util.DateTime` in a value with Date type.
`CDateFromUnoTime()`	Convert a UNO structure `com.sun.star.util.Time` in a value with Date type.
`CDateToISO()`	Return the ISO formatted date (`YYYYMMDD` from a value with Date type. 7/14/2017, `CDateToISO(Now())` → `"20170714"`
`CDateToUnoDate()`	Return the date in a UNO structure `com.sun.star.util.Date`.
`CDateToUnoDateTime()`	Return the date and time in a UNO structure `com.sun.star.util.DateTime`.
`CDateToUnoTime()`	Return the time in a UNO structure `com.sun.star.util.Time`.
`Date()`	Return the actual date (type Date). See `Now()`, `Time()`
`DateAdd()`	Return a new date calculated from a start date and an addition criteria (±). 7/14/2017, `DateAdd("m", 1, Now())` → `8/14/2017` Addition masks `yyyy` year; `q` quarter; `m` month; `y` day of the year; `w` week day; `ww` week of the year; `d` day; `h` hour; `n` Minute; `s` Second
`DateDiff()`	Return the difference between dates, expressed in a specified unit (see table in `DateAdd()`.) `DateDiff("m", "8/14/2017", "7/14/2017")` → `1`
`DatePart()`	Return the specified part of the date (see table in `DateAdd()`). `DatePart("q", "7/14/2017")` → `3`
`DateSerial()`	Return a numerical value for a date, computed from 3 elements year, month and day. `DateSerial(2017,7,14)` →
`DateValue()`	Return a date value from a string representing a date. `DateValue("7/14/2017")` → `07/14/2017` (type Date)
`Day()`	Day of the month. `Day("7/14/2017")` → `14`
`Hour()`	Hour. If noon. `Hour(Now())` → `12`
`Minute()`	Minutes of a date in Date type. If noon. `Minute(Now())` → `0`
`Month()`	Month number. `Month("7/14/2017")` → `7`
`Now()`	Return current date and time (Date type). See `Date()`, `Time()`
`Second()`	Seconds of a date in Date type. If noon. `Second(Now())` → `0`
`Time()`	Return current time (Date type). See `Date()`, `Now()`
`Timer()`	Return a double indicating the number of seconds from midnight. ☞ Assign `Timer()` to a variable before using it !
`TimeSerial()`	Return a date value for a time, computed from 3 elements hour, minute and second. `TimeSerial(12,25,14)` → `12:25:14` (type Date)
`TimeValue()`	Return a time value from a string representing a time. `TimeValue("12:25:14")` → `12:25:14` (type Date)
`Wait`	(instruction) Wait a specific number of milliseconds. `Wait 1000` → pause of 1 sec.
`WeekDay()`	Day of the week (`1` = Sunday). `Weekday("7/14/2017")` → `6` (Friday)
`Year()`	Year number. `Year("7/14/2017")` → `2017`

Color functions

Color are stored in `Long`.

`Red()`, `Green()`, `Blue()` Extract the color components.

`RGB()` Return a color number from red, green, blue components.
`RGB(128,0,0)` → `8388608` (red)

Array functions

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

`DimArray()` As `Array()`: `MyArray = DimArray("One", 2, Now())`
☞ Use when implicit declaration of variables. Otherwise, use `Array()`.
`Erase()` (Statement) Deletes array contents. If the array is dynamic, frees memory. `Erase(MyArray)`
`LBound()` Lower bound
`UBound()` Upper bound

Variables information functions

These functions return information on variables.

On every variable

`TypeName()` Return a string with details on a given variable.
`VarType()` Return a numerical value related to a given variable.
`IsUnoStruct()` Return True if argument is a UNO structure.

The first 2 functions return values listed below:

VarType	TypeName	VarType	TypeName	VarType	TypeName
0	Empty	7	Date	16	Char
1	Null	8	String	18	UShort
2	Integer	9	Object	19	ULong
3	Long	11	Boolean	20	Long64
4	Single	12	Variant	35	INT64
5	Double	17	Byte		
6	Currency	37	Decimal		

Withe greyed background, internal types not used, in principle.

On the variants

Return `True` according to the real detected type.

Function	Verify type	Function	Verify type
`IsArray()`	Array	`IsNull()`	`Null` (no data).
`IsDate()`	Date.	`IsNumeric()`	Numerical value.
`IsEmpty()`	Non initialized variable.	`IsObject()`	OLE object.
`IsError()`	Error value.		

On objects

`HasUnoInterfaces()` `True` inf an UNO object supports interfaces
`(obj.)SupportsService()` `True` if object `obj` supports the service passed as argument (between quotes).

Typecast functions

These function convert value of a given type into another. The function name reflects target type.

☞ Code reading: always prefer an explicit typecast instead of an implicit typecast !

`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)

Miscellaneous functions

`GetGUIType()` Return a value specifying the system, among:
1 Windows
3 MacOS
4 OSX or Linux
`GetSolarVersion()` Return LibreOffice version.
`IsMissing()` Test if an optional parameter of a Sub is omitted.

Call system commands

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

`Command` Command to execute (`String`).
`Style` Widows to execute command, among (`Integer`):
- `0` The program receives focus and its windows is masked.
- `1` The program receives focus and is launched in a standard-sized window.
- `2` The program receives focus and is launched in a minimized window.
- `3` The program receives focus and is launched in a maximized window.
- `4` The program starts in a standard window without receiving focus.
- `6` The program starts in a minimized window; focus remains in current window.
- `10` The program starts in full screen mode.

`Param` Command execution parameters (`String`).
`Synchro` Execution flag:
- `True` Wait until command execution ends.
- `False` Do not wait until command execution ends.

Format function - Formatting masks

A **formatting mask** is a string that can be divided in 3 sections separated by semi-colons: `val>0;val<0;val=0`. One single section = all numbers.

☞ For localized number formatting, check **Tools > Options > Language Settings > Language**.

Numbers

`0`	Mandatory digit (otherwise 0)	`%`	Display results in percents.
`#`	Optional digit.	`E- E+` `e- e+`	Scientific format
`.`	Decimal separator		
`+- space ( )`	Literals, as in the results.	`\`	Escaping char. : the following char. is displayed in the result.

Dates

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

Error handling

Error information

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

VBA Support

☞ Support for VBA is not complete.

Environment options

Tools > Options > Load/Save > General

Load Basic code Loads and saves VBA code of a Microsoft Office document in a special LibreOffice Basic module.
Executable code The VBA code will be loaded for execution.
Save original Basic code VBA code is stored back into file.

Runtime options

VBA support requires: `Option VBASupport 1` and `Option Compatible`.

VBA Functions

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

More information in LibreOffice Help

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. If 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 de Chartres [attr.])

History

Version	Date	Comments
1.01	October 1st 2017	First version.

Licence

This reference card is under license
CreativeCommons BY-SA v3.
Information
https://creativecommons.org/licenses/by-sa/3.0