

# APPENDIX TO THE De Re BASIC! MANUAL

(Version 1.91, 2017-03-14)

ENHANCEMENTS and CORRECTIONS as

# OliBBasic

# 3.00

# APPENDIX TO THE MANUAL

(De Re BASIC! Version 1.91, 2017-03-14)

## ENHANCEMENTS and CORRECTIONS

A fork called OliBasic 3.00 based on RFO-BASIC! 1.91 2022-03-31gt

Formatted like the manual, changes and corrections colored in dark Magenta  
the newest in turquoise.

What should you keep in mind, if you use source code that runs with BASIC! 1.91 perfectly.

1. The minSDK is 19, the targetSDK is 29
2. Look at FILE.EXISTS the second parameter is now a value or a String expression!
3. \_\_\_\_\_
4. The automatic low memory warning is switched off. Use instead OnLowMemory:.
5. The Global Value Backdoor, after an interrupt is trapped, is now closed.  
If getting issues in conjunction with functions in older code  
created for RFO-Basic 1.91 use:  
ON\*\*\*:  
  GLOBALS.ALL  
  
  ....  
  GLOBALS.NONE  
\*\*\*.RESUME
6. The RUN command has new possibilities.
7. AndroidManifest.xml has the new permission ACCESS\_NETWORK\_STATE.
8. This version use absolute paths and relative paths.
9. An internal directory is created at the first start.
10. Add more system constants at FILE.ROOT
11. FILE.EXISTS returns readable and writeable infos, too.
12. FILE.LASTMODIFIED is new.
13. FILE.EXISTS accepts string expressions and values. See description.
14. BUNDLE.GET and DEBUG.DUMP.BUNDLE convert Java Objects to strings.
15. BUNDLE.TYPE modified
16. BigDecimal implemented
17. ROUND() rounding option Legacy int() added
18. BYTE.COPY append switch added
19. SOCKET.SERVER.READ.BYTE is new
20. SOCKET.CLIENT.READ.BYTE is new
21. Scheduler commands implemented
22. Dialog.message, Dialog.select and Select can now be finished at a given time interval
23. At launching with a given program path and an Intent Extra with a key named  
  "\_BASIC!" and a String expression "\_Editor" the program starts in the Editor mode.
24. BUNDLE.OUT is new.
25. On Runtime Error a Broadcast is send.
26. Editor improvement: Switching automatic word completions or -corrections to off.

## ATTENTION

***All result arguments of the BigDecimal commands are on the left side now.  
I'm so sorry for this inconvenience!***

- 27. EMAIL.SEND extended with CC, BCC, Attachments ...
- 28. AUDIO.INFO is new.
- 29. AUDIO.RECORD.PEAK is new.
- 30. AUDIO.RECORD.START has new options.
- 31. AUDIO.LOAD a http stream can also be loaded.
- 32. AUDIO.PLAY second parameter for output channel(s)
- 33. AUDIO.VOLUME third parameter for outer volume control
- 34. DIR, FILE.DIR, FTP.DIR and ZIP.DIR have a time stamp option now.
- 35. GR.BITMAP.PUT replaced Bundle.PP
- 36. GR.BITMAP.GET replaced Bundle.GP
- 37. TEXT.INPUT improvements: Opportunity to switch off the text suggestion, Menu
- 38. USING\$() needs for %d %o %x %X and %t no more Int().
- 39. LIST.SORT is new. It sorts the contents of the given list, as an option also in language and country specific order.
- 40. REPLACE\$() has a new modifier using regular expressions.
- 41. LIST.SPLIT is new. It splits a list into two new lists item by item.
- 42. LIST.JOIN is new. It joins two lists into a new list item by item.
- 43. LIST.MATCH is new. Returns a match found for a search argument in a list.
- 44. Delimiter and number-of-colums enhancements for SQL.NEW, SQL.INSERT and SQL.UPDATE. It takes a little more effort because legacy reasons, but it is needed if you want to import different tables automatically.
- 45. GR.OPEN improvement: Camera view in background, sorry only Android 6+ today
- 46. GR.CAMERA.FOCUS is new in graphics mode in conjunction with GR.OPEN.
- 47. GR.CAMERA.FLASH is new in graphics mode in conjunction with GR.OPEN.
- 48. GR.CAMERA.ZOOM is new in graphics mode in conjunction with GR.OPEN.
- 49. GR.CAMERA.GETPARAM is new in graphics mode, sorry only Android 5+ today
- 50. GR.CAMERA.SETPARAM is new in graphics mode, sorry only Android 6+ today
- 51. GR.CAMERA.DIRECTSHOOT is new in graphics mode in conjunction with GR.OPEN.
- 52. Other GR.CAMERA. \*\*\* SHOOT commands enhanced with more arguments
- 53. The SELECT list view can be controlled via the layout parameter bundle now.
- 54. DIR and FILE.DIR also have access to assets now.
- 55. CONSOLE.ORIENTATION is new. Works like GR.ORIENTATION
- 56. CONSOLE.LAYOUT is new.
- 57. CONSOLE.DEFAULT is new.
- 58. CONSOLE.LINE.TOUCHED and SELECT also return a double touch tap now.  
To save execution time the following GR commands are a little optimized.
- 59. GR.SCALE supports as a new option a translation in X and Y direction.
- 60. GR.SCALE.TOUCH is new and the opposite of GR.SCALE.  
*Multi-touch with more than two touches are supported. New commands needed.*
- 61. GR.ARRAY.TOUCH is new and returns pairs of touch coordinates as an array.
- 62. GR.LIST.TOUCH is new and returns pairs of touch coordinates as list(s).
- 63. GR.LAST.TOUCH is new and returns the last touch index and coordinates.
- 64. BYTE.OPEN allow by an internal cache access to converted assets files now.
- 65. BYTE.COPY break switch added, making an abort by an interrupt possible.
- 66. SHELL is new. Sends a shell command to the system and wait for a result.
- 67. SENSORS.LIST is fixed and features new optional sensor infos (Android 5+).
- 68. KB.SEND.KEYEVENT is new. Sends key events to the environment.

69. Some USB or Bluetooth keyboard function keys are supported in the Editor now.
70. STT.LISTEN has a new bundle argument to control more options.
71. BACKGROUND checks if the display screen is off and if the device is locked now.
72. FTP.OPEN new opportunity to use FTPS, also.
73. GLOBALS.FNIMP is new. It imports a variable from the main code into a function.
74. ARRAY.BINARY.SEARCH is new. Very fast!
75. LIST.BINARY.SEARCH is new. Very fast!
76. GR.CAMERA.TAKEVIDEO is new.
77. More Editor enhancements, sub menu, and options for code pre handling with  
Basic programs
78. BYTE.WRITE.BUFFER improved for more speed.
79. BYTE.READ.BUFFER, BYTE.WRITE.BUFFER, GRABFILE, GRABURL have or  
have more options to select a character set now.
80. INKEY\$ returns the UTF-8 character also.
81. BIGD.FROMDOUBLE is new.
82. BIGD.TODOUBLE is new.
83. BIGD.FROMBASE like BIN, HEX, OCT, but an Integer part greater than 15  
digits is possible.
84. BIGD.TOBASE like BIN\$, HEX\$, OCT\$, as an opposite to BIGD.FROMBASE.  
*The special Floating Point numbers NaN (Not a Number) and Infinity get better support now.*
85. IS\_NAN is new and detects NaN (Not a Number) Floating Point numbers
86. IS\_INFINITE is new and detects Infinity Floating Point numbers
87. WITHIN() is new. Logical function to search in geometry figures.
88. REVERSE\$() is new. Also known as MIRROR\$() in other Basic dialects.
89. GR.CLS has a new option to delete all used bitmaps now.
- 90. If it is possible, bitmaps will be overwritten now. That have some  
consequences. See Bitmap Commands / Overview**
91. CONSOLE and SELECT are able to print text and images with HTML tags now.
92. New selection opportunities for CONSOLE and SELECT.
93. CLAMP() is new.
94. LIST.JOIN added \_min and \_max
95. GR\_COLLISION New option for setting a distance collision border.
96. GR.BEHIND, GR.INFRONT, GR.TOBACK and GR.TOFRONT are new.
97. Improvements in speed and power consumption associated with console output
98. BT.UTF\_8.READ and BT.UTF\_8.WRITE are new. Transfer full UTF-8 char. set.
99. UDP.READ and UDP.WRITE are new and support the UDP protocol.
100. HTTP.REQUEST is new and enhances with more request types and options.
101. GR.COLOR supports also Porter-Duff masking now.
102. GR.PAINT.RESET resets Porter-Duff also.
103. PROGRAM.INFO now extended with memory information.
104. SENSORS.READ supports sensors which returns more than 3 parameters.
105. In the Editor's SubMenu item Previous is new. You can choose one of the ten  
different last loaded Basic files.
106. APP.SAR supports createChooser().
107. NOTIFY is enhanced.
108. NOTIFIY.CANCEL is new.
109. CONSOLE.TITLE is enhanced with subtitle, icon and a hiding option  
**Note that the console title must be set manually now. Thus, no title  
"BASIC! ... Program Output" is displayed at runtime.**
110. Changes in the Editor's Action Bar for more program name space.

111. GR.BITMAP.DRAWINTO.END now has an option to suppress a runtime error
112. APP.SETTINGS opens the application settings
113. DEVICE.OS is new and needs **no** Phone permissions.
114. Permission description and commands
115. FILE.ABSOLUTE is new and returns the absolute file path.
116. IS\_GR() is new and returns the graphic mode status.
117. SELECT is enhanced with subtitle, icon and a hiding option
118. Drawables are now supported. (**Animated** Drawables, Android 9+)
119. Broadcast messages are supported, but forgotten to insert here.
120. GR.SCREEN is extended
121. ONGRSCREEN and GR.ONGRSCREEN.RESUME are new.
122. APP.SAR supports getBooleanExtra() now.
123. JSON support
124. XML support in conjunction with JSON
125. BUNDLE supports Booleans, Drawables and JSON now.
126. SPC\$ is new. It returns a number of spaces as a string.
127. Android 9 cut out (notches) support in graphic mode.
128. GR.OPEN enhancement, a translucent or hidden navigation bar.
129. GR.SET.ACCELERATION is new.
130. ARRAY.RND is new, which creates arrays with random numbers.
131. ARRAY.TO.DIMS is new, which copies and re-dimensions existing arrays.
132. SQL.SET\_LOCALE is new, set the current locale for a specified SQL database
133. SQL.PING is new, returns information about a SQL database.
134. REDIM is new, Re-Dimensions an existing array
135. Array copy by B[ ] = A[ ] is new
136. Functions are able to return arrays now.
137. GR.HIDE and GR.SHOW support multiple arguments now.
138. GR.BITMAP.CLR is new and fills a bitmap complete with transparency
139. PROGRAM.INFO returns also true if a program is started by a launcher now.
140. ONGRTOUCHMOVE:, ONGRTOUCHUP: and their Resume counterparts are new
141. GR.CLIPOUT is new and compensates for the new limitations of Gr.clip.
142. DEVICE.USB is new and returns the parameters of plugged in USB devices.
143. CONSOLE.LINE.TOUCHERD returns swipes now.
144. A lot of decor stuff for Status, Action and Navigation Bars like off, on, colors, menus ...
145. ONMENUITEM:, MENUITEM.RESUME and MENUITEM.GET.DATALINK are new.
146. ONHTMLRETURN: and HTML.ONHTMLRETURN.RESUME are new.
147. FILE.ABSOLUTE converts Document Paths into File Paths if possible now.
148. ADOC command group is new and takes care about Android Documents.
149. IS\_HTML() is new and returns the HTML mode status
150. HTML.TO\_PDF is new and returns a HTML view into a PDF document.
151. HTML.PAPERFORMATS is new and returns a bundle with possible paper formats
152. FILE.DIR extended with recursive results
153. GR.BITMAP.FILTER is new and provides bitmap filters
154. Between ON\*: and \*.resume Format inserts also spaces.
155. GR.SET.CAP is new and set the line caps.
156. GR.PAINT.SET is new and provides Shaders as an example.
157. LIST.MAP.2D is new and maps translations, rotations and multiplications.
158. LIST.MAP.3D is also new and maps 3D operations in a 3D space.
159. Editors's Load and Delete file lists can be sorted by Date in reverse order now.
160. GR.BITMAP.SIZE allows direct file access and SVG files also.
161. GR.BITMAP.LOAD is enhanced with SVG, cropping and a background color.

- 162. FILE.MD5 is new and returns the MD5 hash of a file.
- 163. HEX\$() is enhanced and accepts a color definition string as input also.
- 164. ARRAY.MATH is new and increases the speed significantly.
- 165. LIST.SORT.BY is new and returns an index List, a sorted List by a given List.
- 166. GR.BITMAP.GET.PIXARR is new and returns the bitmap content as Arrays.
- 167. GR.BITMAP.SET.PIXARR is new and returns a bitmap specified by Arrays.
- 168. GR.BITMAP.GET.HISTOGRAM is new and returns color channel histograms.
- 169. ARRAY.MAT.TRANSPOSE is new and transposes a two-dimensional Array.
- 170. ARRAY.MAT.SKILL is new and supports a lot Matrix and Array skills.
- 171. ARRAY.FROM.STRING is new and returns a numeric Array of character codes
- 172. ARRAY.TO.STRING is new and returns a String created by a numeric Array
- 173. ARRAY.MAT.TOGGLE is new and toggles between column or row interpretation
- 174. ZIP.FILES is new and can zip a lot of files by one command call.
- 175. ZIP.EXTRACT is new and can extract complete directories by one command call.
- 176. FILE.COPY is new and copies files and directories. (Android 8+)
- 177. FILE.MOVE is new and moves files and directories. (Android 8+)
- 178. SQL.CCL is new and clears the global SQL Cursor List.
- 179. GR.POLY has a new option. The Polygon can be open.
- 180. AUDIO.RECORD.START supports also PCM 16BIT \*.wav files now.
- 181. AUDIO.RECORD.BUFFER is new and returns a PCM 16BIT buffer.
- 182. ARRAY.TRUTH.SUBSET is new and returns a subset of true or false members.
- 183. ARRAY.TRUTH.INDEX is new and returns a subset of true or false indexes.
- 184. GR.TARGET.MODIFY is new and works directly with Arrays.
- 185. FILE.DELETE is enhanced with recursive deleting child files and directories.
- 186. FILE.SET.LASTMODIFIED is new and overwrites the date of the last modification.
- 187. GR.TEXT.WRAP is new and wraps the text within a given pixel width.
- 188. ONEX\$() is new and returns the ordinal number extension of a given numeric value.
- 189. GR.SET.DASHPATHEFFECT is new and sets a dash effect for line paths.
- 190. GR.PATH is new and creates a path with lines and curves.
- 191. FORMAT\$() can now use numbers which are stored as a String (BigD).
- 192. FTP commands have an option to return success and errors now.
- 193. GOTO.GET.ERROR.INDEX is new and returns the current execution index.
- 194. GOTO.GET.INDEX is new and returns the current execution index.
- 195. GOTO.SET.INDEX is new and jumps to the given execution index.
- 196. APP.INFO is new and returns information about an app.
- 197. PROVIDER is new and manages File and Adoc Provider.
- 198. UCODE32() detects 16- and also 32-bit characters.
- 199. FILE.SELECT is new and provides a directory and file browser.
- 200. DIALOG.SELECT is enhanced with buttons.
- 201. DIALOG.MESSAGE is enhanced with layout options.
- 202. NFC.READ and NFC.WRITE are new and support NFC tags and cards.
- 203. FILE.ROOT.SET.DATA is new and changes the default data path.
- 204. FILE.ROOT.SET.DATABASES is new and changes the default database path.
- 205. FILE.ROOT.RESET is new and resets changed data and database paths.
- 206. If OliBasic will be new installed the Base Directory is in the internal protected path.
- 207. The Base Directory can be changed by the Preferences menu.
- 208. A FTP server is now part of the IDE.
- 209. CLIPBOARD.INFO is new and returns a description of the clipboard content.
- 210. CLIPBOARD.GET and CLIPBOARD.SET are enhanced.
- 211. BUNDLE.GET, BUNDLE.PUT and BUNDLE.CONTAIN accept multiple keys now.



212. DIALOG.MULTI is new and provides a multiple choice dialog.  
 213. DIALOG.SINGLE is new and provides a single choice dialog with pre-selection.  
 214. DIALOG.CUST.\* commands are new and offer customized dialogues.  
 215. JOIN(.ALL) is enhanced with the possibility of numeric arrays.  
 216. NTRIM\$ is new and trims a numeric value to the smallest possible string length.  
 217. GR.ARCPOLY is new and creates a polygon of arcs and lines.  
 218. STT.LISTEN is enhanced with a hidden mode and non execution.  
 219. COLOR() is new and returns the system color number of a color string expression.  
 220. COLOR\$() is new and returns a color string expression of a system color number.  
 221. GR.BITMAP.GET.SELECTED.PIXARR is new and returns colors at specified pixels.  
 222. FOR accepts index and Array now.  
 223. MESH.HULL is new and returns the hull of a 2D point cloud.  
 224. QR.CREATE.SVG is new and creates a SVG file with a QR code.  
 225. FILTER group is new and provide filters like FFT.  
 226. CLOCK() enhanced with a possibility of nanoseconds.  
 227. GR.GET.BOUNDS is new and returns the bounds of the most graphic objects.  
 228. LIST.BOUNDS.2D is new and returns the bounds of a xy List.  
 229. LIST.BOUNDS.3D is new and returns the bounding box of a xyz List.  
 230. GR\_COLLISION object type Line added.  
 231. WITHIN() new option to compare two xy Lists.  
 232. FTP.SERVER commands to control the build-in FTP-Server are new.  
 233. ARRAY.MEDIAN is new and returns the median of an Array.  
 234. EVEN () is new and checks the given value.  
 235. ODD () is new and checks the given value.  
 236. FILTER() is new and support some filter types.  
 237. FILTER.SET is new and sets filters.  
 238. Enhanced ARRAY.MATH with Odd, Even and Filter  
 239. MESH.TRIANGLE, MESH.TRIANGLE.MIDPOINT and MESH.TRIANGLE.2.5D, new  
 240. Enhanced GR.POLY with multiple polygons and multiple Gr.Paints.  
 241. Enhanced GR.COLOR with HSV color spectrum including the Hue color wheel.  
 242. HTML.EVALUATE.JS is new and sends a JavaScript content to evaluate.  
 243. HTML.GET.DATALINK enhanced with the result of HTML.EVALUATE.JS.  
 244. HTML.GET.DATALINK enhanced with Console Message and STT.  
 245. MESH.STL.LOAD and MESH.STL.SAVE are new to handle binary STL files.  
 246. GR.PAINT.LIST is new and creates a Paint pointer List by a color string List.  
 247. LIST.SPLIT.2D and LIST.SPLIT.3D are new and split xy(z) Lists in their components.  
 248. LIST.JOIN.2D and LIST.JOIN.3D are new and join List components into xy(z) Lists.  
 249. LIST.CREATE accepts more than one List of the same type now.  
 250. LIST.CLEAR clears more than one List now.  
 251. LIST.GET returns more than one value if wished now.  
 252. LIST.REPLACE replaces more than one value if wished now.  
 253. LIST.REMOVE removes optionally more than one item now.  
 254. LIST.ADD.LIST is also able to add a sub List now.  
 255. LIST.ROW.PRINT is new and prints a List as rows into the console or a String.  
 256. ARRAY.ROW.PRINT is new and prints an Array as rows into the console or a String.  
 257. Additional comment signs: // - Single Line and - Middle of Line; /\* and \*/ - Block  
 258. LIST.KILL.LAST, STACK.KILL.LAST and BUNDLE.KILL.LAST are new.  
 259. ARRAY.TRUTH.CHOICE is new to build array structures of IF, ELSEIF, ELSE.  
 260. LIST.TARGET.MODIFY is new and supports some targets like Gr.target.modify.  
 261. FILE.REPLACE is new and replaces a String in a file or files within directories.  
 262. LIST.SPREAD is new and spreads the contents of an array across lists evenly.

- 263. LIST.DIMSORT.BY is new and can sort dim-ed sources by dim-ed sort sources.
- 264. FILE.SHA is new and returns the SHA hash of a file.
- 265. BLE command group is new and supports Bluetooth Low Energy (BLE).
- 266. Enhancements around Bt.status like OnBtStatus: and Bt.onStatus.resume.
- 267. USB command group is new and supports serial data transfer via USB.
- 268. AUDIO.INFO is extended by a few additional arguments.
- 269. BT.OPEN enhanced with delimiter options.

Some fixes



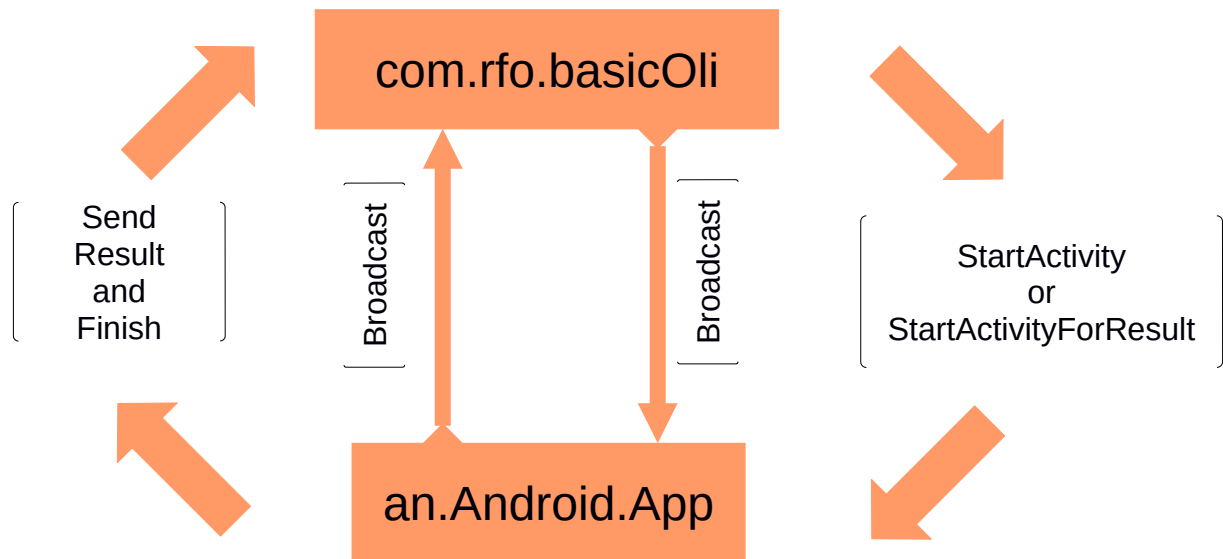
Thanks to Alberto, Andrey, Bob, Brochi, [Carlos](#), Chris, Emile, [Guillermo](#), Humpty, Janusz, [Jean-Manuel](#), [Jürgen](#), [knoWare](#), Mog, Michael, Nicolas, Roy, Spike for testing and support. Special thanks to Paul for creating RFO-Basic and Marc for the great enhancements and criticisms with substance.

Special thanks [also](#) to:

- Emile for text corrections
- Humpty for the source of Globals.fnimp [and other great stuff I have taken over](#)
- Janusz for the hints of the needs of visually impaired users
- Stephen C (Stackoverflow) about "What About Reading Unknown Number of Bytes?"
- Brochi for the bit shifting code and other wonderful hints and code examples
- Guillermo for testing display cut outs
- KnoWare for the SQL know-how
- Spike for creating a knowledge base wiki
- Jürgen Strambach for a lot of excellent examples and testing new stuff
- Paul LeBeau for best support about SVG cropping
- Moritz Stückler (ftpServer)
- Project Nayuki (FFT, QR code, Point cloud hull)
- Bernd Rost for code and help in conjunction with filters
- Felipe Herranz for his USB interface

Happy coding  
Gregor Tiemeyer

## The Closed Process Circle



**App.installed <flag\_nexp>,**

**<package\_sexp>{{{,<versionName\_sexp>,<versionCode\_nexp>,<pmRaw\_sexp>}}**

If the android package is installed, flag\_nexp returns 1, else 0. Insert the package id like "com.rfo.basic" or "all.sub.My.APP". VersionName and versionCode are results from the APK manifest. Package manager returns a raw string list in pmRaw\_sexp.

**App.load <package\_sexp>**

Loads and installs an android app package (\*.apk) with a given parse-able URI.

Example:

```
File.root dataPath$
```

```
! Can be parsed:
```

```
Package$ = "file://" + dataPath$ + "/" + "Bookworm_de.apk"
```

```
Package$ = "market://details?id=" + "com.opera.mini.native"
```

```
! Can not be parsed directly:
```

```
!!begin
```

```
Package$ = "http://mougino.free.fr/tmp/920EditorforBASIC_13720.apk"
```

```
(Download the Apk first to the data path. See also the example which  
can be found in the Sample Program file, f00a_download_manual.bas.)
```

```
Package$ = "file://" + "/android_asset/" + "Bookworm_en.apk"
```

```
(Protected, copy the Apk first to the data path.)
```

```
!!end
```

```
App.Load Package$
```

## APP.SAR <pointer\_nexp>

Start **And** Receive an **App**, **Intent** or Broadcast with a Java like Interface.

The **background** references are cut off!

Eg. FileBrowser, BarCodeScanner, GoogleMaps, ... are supported.

Note: The Blackmoon FileBrowser is suggested, because it does not need the clipboard for multiple files.

There is also APP.START with an other interface and less features (and still an internal design issue), but in some cases an option.

The APP.BROADCAST command does not work properly yet, use BROADCAST (faster) or APP.SAR for complex wishes.

Example:

```
LIST.CREATE S, commandListPointer
LIST.ADD commandListPointer~
"new Intent(Intent.ACTION_GET_CONTENT);" ~
"setPackage(\"com.blackmoonit.android.FileBrowser\");" ~
"setType(\"*/*\");" ~ %From Ex.: intent.setType( "*/*");
"addCategory(Intent.CATEGORY_DEFAULT);" ~
!%From Ex.: intent.addCategory(Intent.CATEG...
"addFlags(Intent.FLAG_ACTIVITY_EXCLUDE_FROM_RECENTS);" ~
! → );" ~ ← is important
"EOCL"

LIST.CREATE S, resultListPointer
LIST.ADD resultListPointer~
"theFilePath$=getData().getPath();" ~
"EORL"

BUNDLE.CREATE appVarPointer
BUNDLE.PL appVarPointer,"_CommandList", commandListPointer
BUNDLE.PL appVarPointer,"_ResultList", resultListPointer
BUNDLE.PUT appVarPointer,"theFilePath$", "No file selected!"

APP.SAR appVarPointer
```

ATTENTION: In Java variable, class and function names are case sensitive. In most Java examples → **intent**. ← is a variable of the class Intent and have to be deleted.

Instead → **Intent**. ← is the class and strongly not to delete.

**\_CommandList** is a required key expression.

**\_ResultList** is a required key expression if you want results.

**\_Broadcast** is a required key expression if you want to send a Broadcast.

Basic! Variables used in the CommandList (only the bracketed variables with exclamation like "!variable\$!") and ResultList have to be put in the transfer bundle, too.

Ignore and delete → this. ←.

Note: The maximum size for the intent bundle is limited to about 1 MB when transferring data with intents.

TIP: If the expected results is not returned then PRINT the expressions first to debug your code.

Supported:

```
new Intent
setAction
setPackage
putExtra only Bundle, String, Long, Double and the arguments true and false
addCategory
```

```

addFlags
setComponent(new ComponentName(
setData          Caution: If you want to set both the URI and MIME type, don't call setData() and
setDataAndType  setType() because they each nullify the value of the other. Always use
setType          setDataAndType() to set both URI and MIME type.

```

```

getStringExtra
getIntExtra
getDoubleExtra
getBundleExtra
getBooleanExtra
getData().getPath()
getLaunchIntentForPackage()
getParcelableArrayListExtra(Intent.EXTRA_STREAM)

```

```

Out of The box
createChooser
"createChooser("+ CHR$(34) + "Chooser Title, share with ..." + CHR$(34) + ");" ~

```

This list will maybe expanded. Stay tuned!

In the underlying Java machine were two main types of application starts implemented.

First type: **startActivity** used by App.Start and App.ASR without \_ResultList

In this case the activity is **not launched** as a **sub-activity**!

Second type: **startActivityForResult** used by App.ASR with \_ResultList

Important: If you apply App.ASR with \_ResultList use the suggested ReorderToFront() function below before finishing, which prevents trouble by hitting the home button at sub intent runtime. In this case Console.front and Gr.front fail.

If the called intent was written in Java or an other language as BASIC!, use the appropriate commands.

A look at <https://developer.android.com/guide/components/intents-filters> is recommended.

See also Bundle.In, Bundle.Out

Examples:

```

!----- ReorderToFront() also part of TestTest.bas -----
FN.DEF ReorderToFront()
PROGRAM.INFO b
BUNDLE.GET b, "_PackageName", pN$
LIST.CREATE S, commandListPointer
LIST.ADD commandListPointer~
"new Intent(Intent.ACTION_MAIN);" ~
"setPackage("+ CHR$(34) + pN$ + CHR$(34) + ");" ~
"addCategory(Intent.CATEGORY_DEFAULT);" ~
"addFlags(Intent.FLAG_ACTIVITY_REORDER_TO_FRONT);" ~
"EOCL"
BUNDLE.PL appVarPointer, "_CommandList", commandListPointer
APP.SAR appVarPointer
FN.RTN appVarPointer
FN.END
!----- Rest of TestTest.bas -----

```

```

PROGRAM.INFO bPointer
BUNDLE.GET bPointer, "_BasName", bN$
BUNDLE.GET bPointer, "_PackageName", pN$
BUNDLE.GET bPointer, "_MyProcessId", mpid$
CONSOLE.TITLE bN$ + " runs on the " + pN$ + " Basic Engine"
? "_MyProcessId", mpid$
file.root fp$,"_Programpath"
?fp$
BUNDLE.IN action$, data$, bi
?"action$ ";action$
? "data$ "; data$
p = 2000
? "Waiting " + int$(p/1000) + " seconds before killing"
PAUSE p
BUNDLE.PUT resultPointer, "EXTRA_RESULT", "EXTRA_RESULT Strings Test"
BUNDLE.PUT resultPointer, "Test", "Strings Test"
BUNDLE.PUT resultPointer, "TestNum", 4711
BUNDLE.OUT a$, d$, resultPointer
!To prevent a Home-button-hitting-issue if you need automatic itself-returning
ReorderToFront()
EXIT

```

If you use in the next examples the variable basicEngine\$ you can use:

```

com.rfo.basicOli ,
com.rfo.basicFellow, NEW
com.rfo.basic, with limitations

```

Or crate your own one with:

```

BUNDLE.IN action$, data$, bi
FILE.EXISTS ok, data$
IF ok THEN RUN data$
sel = -4000 %NEW If negative the dialog.message will close after <sel> Milliseconds
DIALOG.MESSAGE "Program File not Found!", data$, sel
EXIT

```

If you call a second BASIC! engine start maybe with com.rfo.basicFellow from OliBasicFellow\*\*.apk and call a **different** engine maybe com.rfo.basicOli.

```

!----- SubIntentWithResult.bas -----
FN.DEF SubIntentWithResult(basEngine$, basProgramPath$, mode, testMode)
eMode$ = ""
IF mode > 0 THEN eMode$ = "_Editor"
LIST.CREATE S, commandListPointer
LIST.ADD commandListPointer~
"new Intent(Intent.ACTION_MAIN);" ~
"setData("+ CHR$(34) + basProgramPath$ + CHR$(34) + ");" ~
"new ComponentName(\"\" + basEngine$ + "\"\" + ","\"\" + basEngine$ + ".Basic" + "\"\"+");" ~
"addCategory(Intent.CATEGORY_EMBED);" ~
!Starts program in Editor mode, if eMode$ = "_Editor"!
"putExtra("+ CHR$(34) + "_BASIC!" + CHR$(34) + "," + CHR$(34) + eMode$ + CHR$(34) + ");" ~
"addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);" ~
"EOCL"

LIST.CREATE S, resultListPointer
IF testMode > 0 THEN
LIST.ADD resultListPointer~
"theDataPath$=getData().getPath();" ~
"EXTRA_RESULT$=getStringExtra("+ CHR$(34) + "EXTRA_RESULT" + CHR$(34) + ");" ~
"Test$=getStringExtra("+ CHR$(34) + "Test" + CHR$(34) + ");" ~
"TestNumN=getDoubleExtra("+ CHR$(34) + "TestNum" + CHR$(34) + ");" ~
"EORL"
ELSE
LIST.ADD resultListPointer~
"EORL"
ENDIF

BUNDLE.PL appVarPointer,"_CommandList",commandListPointer
BUNDLE.PL appVarPointer,"_ResultList",resultListPointer
IF testMode > 0 THEN
BUNDLE.PUT appVarPointer,"theDataPath$",basProgramPath$
BUNDLE.PUT appVarPointer,"EXTRA_RESULT$", EXTRA_RESULT$
BUNDLE.PUT appVarPointer,"Test$", Test$
BUNDLE.PUT appVarPointer,"TestNumN", TestNum
ENDIF

APP.SAR appVarPointer
FN.RTN appVarPointer
FN.END
FN.DEF ReturnedItems$(appVarPointer, pr)
BUNDLE.KEYS appVarPointer, keyListPointer
LIST.SIZE keyListPointer, listSize
IF pr
? "listSize " , listSize-4
FOR I = 1 TO listSize
LIST.GET keyListPointer, I, item$
IF item$ = "_CommandList" | item$ = "_ResultList" | item$ = "%succeededResults%" ~
| item$ = "DebugSI"
ELSE
PRINT item$
ENDIF
NEXT i
ENDIF
BUNDLE.GET appVarPointer,"%succeededResults%", Results$
FN.RTN Results$

```



```

FN.END
basEngine$ ="com.rfo.basicOli" % Your favorite BASIC! engine
FILE.ROOT fp$ , "_External" % Start point of the public part of the external file system
basProgramPath$ = "file://" + fp$ + "/rfo-basic/source/TestTest.bas"
mode = 0 %Only execution mode
! mode = 1 %Editor mode
! testMode = 0 % Without results
testMode = 1 % With results
appVarPointer = SubIntentWithResult(basEngine$, basProgramPath$, mode, testMode)
IF testMode
  myReturnedItems$ = ReturnedItems$(appVarPointer, 0)
  PRINT myReturnedItems$
  BUNDLE.GET appVarPointer,"EXTRA_RESULT$",EXTRA_RESULT$
  PRINT "EXTRA_RESULT=: "; "<" + EXTRA_RESULT$ + ">"
  BUNDLE.GET appVarPointer,"Test$",Test$
  PRINT "Test=: "; "<" + Test$ + ">"
  BUNDLE.GET appVarPointer,"TestNumN",TestNum
  PRINT "TestNum=: "; "<";TestNum;">"
ENDIF
END

```

```

!----- IndependentLaunch.bas -----
FN.DEF IndependentLaunch(basEngine$, basProgramPath$, mode)
  eMode$ = ""
  IF mode > 0 THEN eMode$ = "_Editor"
  LIST.CREATE S, commandListPointer
  LIST.ADD commandListPointer~
  "new Intent(Intent.ACTION_MAIN);" ~
  "setData("+ CHR$(34) + basProgramPath$ + CHR$(34) + ");" ~
  "new ComponentName(\""+ basEngine$ + "\"+ \" ;\""+\""+ basEngine$ + ".Basic\" + "\"+\"");" ~
  "addCategory(Intent.CATEGORY_LAUNCHER);" ~
  "putExtra("+CHR$(34) + "_BASIC!" + CHR$(34) + "," + CHR$(34) + eMode$ + CHR$(34) + ");" ~
  "addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);" ~
  ! vv
  "addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);" ~
  "addFlags(Intent.FLAG_ACTIVITY_MULTIPLE_TASK);" ~
  "addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK);" ~
  ! ^^
  "EOCL"
  BUNDLE.PL appVarPointer,"_CommandList",commandListPointer
  APP.SAR appVarPointer
FN.END
basEngine$ ="com.rfo.basicOli" % Your favorite BASIC! engine
file.root fp$ , "_External" % Start point of the public part of the external file system
basProgramPath$ = "file://" + fp$ + "/rfo-basic/source/TestTest.bas"
! mode = 1 % Program starts in BASIC!'s Editor
mode = 0
CALL IndependentLaunch(basEngine$, basProgramPath$, mode)
END

```

### App.settings {<package\_sexp>}

Calls the application settings of the app with the package-id <package\_sexp>. If the optional package-id is not given, application settings of the running Basic engine or the settings of the executed application created with Basic! is opened.

App.settings is asynchronous on runtime. Use Dialog.message or Dialog.select to halt the program execution.

Example:

```
App.settings % "com.rfo.basicOli"  
Dialog.message "Settings", "Should we check again?", sel, "Yes", "No"  
IF sel = 1 THEN GetPermissions()
```

### App.info <package\_sexp>, <bundle\_pointer\_nexp>

Returns the application information specified by <package\_sexp>. If <package\_sexp> is "" the package\_id of the current application is used.

If you provide a variable that is not a valid Bundle pointer, the command creates a new Bundle and returns the Bundle pointer in your variable. Otherwise it writes into the Bundle your variable or expression points to.

The bundle keys and possible values are in the table below:

Key	Type	Value
_NativeLibraryDir	String	Returns the full path to the directory where application's native JNI libraries are stored.
_PackageName	String	package_id

### Browse <url\_sexp>

If <url\_sexp> starts with "http..." then the internet site specified by <url\_sexp> will be opened and displayed.

If <url\_sexp> starts with "file://" + absolutePath\$ + "/" + fileName\$ then the file will be open by a linked application. Starting with Android 11+ the File Provider is needed for nonpublic files. See PROVIDER.

Note: You can also use the HTML commands to display (and interact with) web pages located on your device or on the web.

See also Gr.camera.takeVideo

**Http.post** <url\_sexp>, <list\_nexp>, <result\_svar>{**{{{**, <ok\_svar>},  
<use\_caches\_lexp>}, <charset\_sexp>}, <connect\_timeout\_nexp>},  
<read\_timeout\_nexp>}

Execute a Post command to an Internet location.

<url\_sexp> contains the url ("http(s)://....") that will accept the Post.

<list\_nexp> is a pointer to a string list which contains the Name/Value pairs needed for the Post. <result\_svar> is where the Post response will be placed.

If the server returns an error if <ok\_svar> is not "ok". An error handler has to be placed behind this command, if needed.

If you want a cache <use\_caches\_lexp> has to be > 0.

To choose a character set use <charset\_sexp>. Default is "\_UTF-8" for text and use "\_ISO-8859-1" for binary data.

If you want to specify a connect timeout use <connect\_timeout\_nexp> with a value > 0.

If you want to specify a read timeout use <read\_timeout\_nexp> with a value > 0.

**Http.request** <request\_type\_sexp>, <url\_sexp>, <list\_nexp>, <result\_svar>{**{{{**,  
<ok\_svar>}, <use\_caches\_lexp>}, <charset\_sexp>}, <connect\_timeout\_nexp>},  
<read\_timeout\_nexp>}

Execute a request command to an Internet location.

Available request types are "\_DELETE", "\_GET", "\_PATCH", "\_POST", "\_PUT" with write and read access and "\_HEAD", "\_OPTIONS", "\_TRACE" only with read access.

<url\_sexp> contains the url ("http(s)://....") that will accept the request.

<list\_nexp> is a pointer to a string list which contains the Name/Value pairs needed for the request. If you have nothing to write, create an empty list and inset this list pointer.

<result\_svar> is where the request response will be placed.

If the server returns an error if <ok\_svar> is not "ok". An error handler has to be placed behind this command, if needed.

If you want a cache <use\_caches\_lexp> has to be > 0.

To choose a character set use <charset\_sexp>. Default is "\_UTF-8" for text and use "\_ISO-8859-1" for binary data.

If you want to specify a connect timeout use <connect\_timeout\_nexp> with a value > 0.

If you want to specify a read timeout use <read\_timeout\_nexp> with a value > 0.

## Home

The **HOME** command does exactly what tapping the HOME key would do. The Home Screen is displayed while the BASIC! program continues to run in the background.

At Android 4.2.1 with 512 MB RAM the BASIC! program does not continue under a lot of circumstances. Beginning with Android 5 tasks are running better in background.

The opposite are the commands Console.front, Gr.front and the suggested ReorderToFront() function. (See examples under APP.SAR) The last one includes all cases inclusive HTML WebView.

### Gr.front flag

Determines whether the graphics screen or the Output Console will be the front-most screen. If flag = 0, the Output Console will be the front-most screen and seen by the user. If flag <> 0, the graphics screen will be the front-most screen and seen by the user.

~~One use for this command is to display the Output Console to the user while in graphics mode. Use **Gr.front 0** to show text output and **Gr.front 1** to switch back to the graphics screen.~~

~~Note: When the Output Console is in front of the graphics screen, you can still draw (but not render) onto the graphics screen. The **Gr.front 1** must be executed before any **Gr.render**.~~

Print commands will continue to print to the Output Console even while the graphic screen is in front.

If you get trouble bringing the application from background to front (Android < 5) use a user defined function like ReorderToFront(). (See examples under APP.SAR) You can use this example for Graphic and Console mode, it returns to the actual used mode automatically.

But there is no guarantee in Android < 5 that the applications returns automatically on top.

You should know, that the Console mode is waiting until the Graphic mode finished. If you use Console.front or Gr.front 0 in Graphic mode **BASIC! will hang**. Use instead maybe Dialog.message, Dialog.select or Select. See also their enhanced command descriptions.

## Console I/O

### Output Console

BASIC! has **four** types of output screens: The Output Console, the Graphics Screen, **and** the HTML Screen **and the Select Screen (last one as a dialog)**. This section deals with the Output Console. See the section on Graphics for information about the Graphics Screen. See the section on HTML for information about the HTML screen.

Information is printed to screen using the Print command. BASIC! Run-time error messages are also displayed on this screen.

There is no random access to locations on this screen. Lines are printed one line after the other.

Although no line numbers are displayed, lines are numbered sequentially as they are printed, starting with 1. These line numbers refer to lines of text output, not to locations on the screen.

**Print {<exp> {,|;}} ...**

**? {<exp> {,|;}} ...**

Evaluates the expression(s) <exp> and prints the result(s) to the Output Console. You can use a question mark (?) in place of the command keyword **Print**.

If the comma (,) separator follows an expression then a comma and a space will be printed after the value of the expression.

If the semicolon (;) separator is used then nothing will separate the values of the expressions.

If the semicolon is at the end of the line, the output will not be printed until a **Print** command without a semicolon at the end is executed.

**Print** with no parameters prints a newline.

Examples:

PRINT "New", "Message"	% Prints: New, Message
PRINT "New"; "Message"	% Prints: New Message
PRINT "New" + " Message"	% Prints: New Message
? 100-1; " Luftballons"	% Prints: 99.0 Luftballons
? FORMAT\$("##", 99); " Luftballons"	% Prints: 99 Luftballons
PRINT "A"; "B"; "C";	% Prints: nothing
PRINT "D"; "E"; "F"	% Prints: ABCDEF

### Print with User-Defined Functions

Note: Some commands, such as **Print**, can operate on either strings or numbers.

Sometimes it has to try both ways before it knows what to do. First it will try to evaluate an expression as a number. If that fails, it will try to evaluate the same expression as a string. If this happens, and the expression includes a function, the function will be called twice. If the function has side-effects, such as printing to the console, writing to a file, or changing a global parameter, the side-effect action will also happen twice.

Eventually this problem should be fixed in BASIC!, but until then you should be careful not to call a function, especially a user-defined function, as part of a **Print** command. Instead, assign the return value of the function to a variable, and then **Print** the variable. An assignment statement always knows what type of expression to evaluate, so it never evaluates twice.

```
! Do this:
y = MyFunction(x)
Print y
```

! NOT this:  
Print MyFunction(x)

### Print with HTML tags for text formatting and image including

With the Console.Layout (also [Select with a layout bundle](#)) command you are able to format your string output in a HTML manner. Thus **colored** letters or included bitmaps may be as bitmap fonts are possible.

### The Output Console and power consumption

The console output is often updated internally. Depending on the device, this process requires more or less power, especially in connection with bitmaps.

Remember to use the SELECT command if possible.

The PAUSE command is also possible, but events are ignored during this period.

Behind the screens:

RFO-Basic1.91 creates a new console list view in its main loop each time, if the console buffer is not empty!

Starting with OliBasicXXII in its main loop, a new console list view is created only if the console buffer is not empty or the hash of the console buffer is changed!

That is, the content of the buffer has changed in this case.

If you use the SELECT command, only a new SELECT list view is created and the main loop is paused until the selection is done with return.

The difference is hot, slightly above body temperature and cool with an older HTC device.

**Btw. if your device body get hot you should replace the battery immediately, because at this point the battery can destroy more than itself.**

Therefore, it is good practice to use the SELECT command instead of the console with all BASIC! versions, if it is possible.

### Console.front

Brings the Output Console to the front where the user can see it.

If BASIC! is running in the background with no screen visible, this command brings it to the foreground. If you have a different application running in the foreground, it will be pushed to the background.

If BASIC! is running in the foreground, but the Graphics or HTML screen is in the foreground, this command brings the Console to the foreground. BASIC! remains in Graphics or HTML mode.

If you get trouble bringing the application from background to front (Android < 5) use a user defined function ReorderToFront() function. (See examples under APP.SAR)

See also Gr.front.

### Console.orientation <nexp>

The value of the <nexp> sets the orientation of screen as follows:

- 1 = Orientation depends upon the sensors.
- 0 = Orientation is forced to Landscape.
- 1 = Orientation is forced to Portrait.
- 2 = Orientation is forced to Reverse Landscape.
- 3 = Orientation is forced to Reverse Portrait.

You can monitor changes in orientation by reading the screen width and height using the **Screen** command.




See also: GR.open, GR.orientation, Select

### Console.layout <layout\_bundle\_nexp>

The layout bundle <layout\_bundle\_nexp> controls the console output layout:

Table of layout control options		
Key	Value	
<b>_TextSize</b>	numeric	
<b>_TextColor</b>	{Alpha,}Red,Green,Blue (comma delimited string) or _{Alpha,}ColorName ({{comma delim.} string) or #{hn}hnhnhn (hex. string)	Note, _TextFont or _TextStyle is needed also!
<b>_TextBackgroundColor</b>	{Alpha,}Red,Green,Blue (comma delimited string) or _{Alpha,}ColorName ({{comma delim.} string) or #{hn}hnhnhn (hex. string)	Has to be "0,0,0,0" if you want a background color, wallpaper or bitmap Note, _TextFont or _TextStyle is needed also!
<b>_TextFont</b>	_Default	
	_Serif	
	_Sans_Serif	
	_Monospace	
<b>_TextStyle</b>	_Normal	
	_Bold	
	_Bold_Italic	
	_Italic	

<b>_TextHtml</b>	0 or 1 (numeric)	<p>Returns displayable styled text from the provided HTML string. But not all tags are supported. Any &lt;img&gt; tags in the HTML will display an image. Absolute ("<a href="#">file:///</a>") and relative paths are allowed. The image size has to be scaled before, because h= and w= are ignored. See _HtmlBitmapScale.</p>  <p>Uses parts of TagSoup library to handle real HTML, including all of the brokenness found in the wild.</p> <pre> &lt;a href="..."&gt; &lt;b&gt; &lt;big&gt;? &lt;blockquote&gt; &lt;br&gt; &lt;cite&gt; &lt;del&gt; &lt;dfn&gt; &lt;div align="..."&gt;? Use instead chr\$(1564) [Arabic Letter] at line begin for align=' right' &lt;em&gt; &lt;font size="..." color="..." face="..."&gt; &lt;h1&gt;, &lt;h2&gt;, &lt;h3&gt;, &lt;h4&gt;, &lt;h5&gt;, &lt;h6&gt; &lt;i&gt; &lt;img src="..."&gt; &lt;p&gt; &lt;small&gt; &lt;strike&gt;? &lt; A.7 &lt;strong&gt; &lt;sub&gt; &lt;sup&gt; &lt;tt&gt;? &lt;u&gt; </pre> <p><b>Replace Space with &amp;#160;; &amp; with &amp;amp;; &lt; with &amp;lt;; &gt; with &amp;gt;; " with &amp;quot;; if necessary.</b></p>
<b>_HtmlTextSelectable</b>	0 or 1 (numeric)	<p>Does only work in conjunction with _TextHtml, but the item selection works only with a <b>long</b> click.</p>

<b>_HtmlBitmapScale</b>	-1, 0, > 0 (numeric)	Does only work in conjunction with _TextHtml. Scales the included bitmaps in the following ways: -1 no scaling, 0 (default) only scaling proportional to the screen resolution > 0 proportional to the font size If it is 1 the bitmap height is the same as the font size.
<b>_DividerColor</b>	{Alpha,}Red,Green,Blue (comma delimited string) or _{Alpha,}ColorName ({comma delim.} string) or #{hn}hnhnhn (hex. string)	
<b>_DividerFilename</b>	bitmap file path	
<b>_DividerHeight</b>	numeric	
<b>_BackgroundWallpaper</b>	0 or 1 (numeric)	Min. Jelly Bean 4.1 (API 16)
<b>_BackgroundColor</b>	{Alpha,}Red,Green,Blue (comma delimited string) or _{Alpha,}ColorName ({comma delim.} string) or #{hn}hnhnhn (hex. string)	
<b>_BackgroundFilename</b>	bitmap file path	Min. Jelly Bean 4.1 (API 16)
<b>_SetSelection</b>	numeric	Sets a pre selected item. The item will not be selected but it will still be positioned appropriately. If the specified selection position is less than 1, then the item at position 1 will be selected.
<b>_StackFromBottom</b>	0 or 1 (numeric)	If 1 pin the view's content to the bottom edge, 0 to pin the view's content to the top edge

For simpleness you can use the same layout bundle for Console.layout and the Select command. But the \_Orientation key will be ignored. In this case also use Console.orientation.

If you want to change something in the layout bundle, it is sufficient to make the changes only in the bundle. Note that every change affects the whole thing. Place a layout change as soon as possible in the Basic! Code. Maybe you have to place a little (PAUSE) break.

See also: Console.orientation, Select

### Console.default

Resets the console layout to the default settings.

### **Console.line.touched** <line\_nvar> {, <touch\_nvar>}

After an **OnConsoleTouch** interrupt indicates the user has touched the console, this command returns information about the touch.

The number of the line that the user touched is returned in the <line\_nvar>.

If the optional <touch\_nvar> is present, the type of user touch a short tap (0), a long press (1) or a double tap (2) is returned in the <touch\_nvar>. The delay for detecting the double tap conforms to the default Android system settings.

If <touch\_nvar> returns numbers greater than 9 it provides swipe directions as follows 10 → right, 11 ↓ down, 12 ← left, 13 ↑ up. If you want to select the line also, keep in mind, that the line has to be high enough. You can set the line height by Console.layout and its key \_ textSize.

It seems, that a long press (1) is on newer Android versions not detectable. Please use instead the SELECT command. It is generally the better choice to select a ListView.

See also: Console.layout, SELECT


**Console.title {{ <title\_sexp>}, <options\_bundle\_nexp>}**

Changes the title of the console window. If the <title\_sexp> parameter is omitted, the title is changed to the default title, "BASIC! Program Output".

**Starting with OliBasic 3.00 the console title is empty by default.**

The optional options bundle <options\_bundle\_nexp> controls the Action and Navigation bar layouts:

Table of layout control options		
Key	Value	Description
<b>_Subtitle</b>	String	Set the action bar's subtitle.
<b>_TitleShow</b>	0 or 1 (numeric)	If 1 (default) Show the Action bar if it is not currently showing. It will resize application content to fit the new space available. If 0 Hide the Action bar if it is currently showing. It will resize application content to fit the new space available.
<b>_TitleIcon</b>	Icon file path	Add a large icon to the notification content view. <a href="http://romannurik.github.io/AndroidAssetStudio/index.html">http://romannurik.github.io/AndroidAssetStudio/index.html</a>
<b>_TitleHomeEnabled</b>	0 or 1 (numeric)	Set whether to include the application home accordance in the action bar. Home is presented as an activity icon. Have to be 1 if you want to show the icon. Have to be 0 if you want to hide the icon. The default setting is API dependent.
<b>_TitleBackground</b>	Background file path	The background of the title bar can be created by a bitmap file. Only one colored pixel is needed.

<b>_TitleHtml</b>	0 or 1 (numeric)	<p>Returns displayable styled text from the provided HTML string. But not all tags are supported.</p>  <p>Uses parts of TagSoup library to handle real HTML, including all of the brokenness found in the wild.</p> <p>&lt;b&gt; &lt;big&gt; &lt;font size="..." color="..." face="..."&gt; &lt;h1&gt;, &lt;h2&gt;, &lt;h3&gt;, &lt;h4&gt;, &lt;h5&gt;, &lt;h6&gt; &lt;i&gt; &lt;small&gt; &lt;strike&gt;? &lt; A.7 &lt;strong&gt; &lt;sub&gt; &lt;sup&gt; &lt;tt&gt;? &lt;u&gt;</p> <p><b>Replace Space with &amp;#160, &amp; with &amp;amp, &lt; with &amp;lt, &gt; with &amp;gt, " with &amp;quot; if necessary.</b></p> <p>Usable for Title and Subtitle. Keep in mind that the ActionBar height will not be expanded.</p>
<b>_StatusbarColor</b>	{Alpha,}Red,Green,Blue (comma delimited string) or _{Alpha,}ColorName (comma delim. string) or #{hn}hnhnhn (hex. string)	Min. Lollipop 5.0 (API 21)
<b>_StatusbarLight</b>	0 or 1 (numeric)	<p>If 0 (default) The Status bar background is dark. In this case the <b>bar content</b> will be <b>light</b>.</p> <p>If 1 The Status bar background is light. In this case the <b>bar content</b> will be <b>dark</b>.</p> <p>Min. Lollipop 5.0 (API 21)</p>



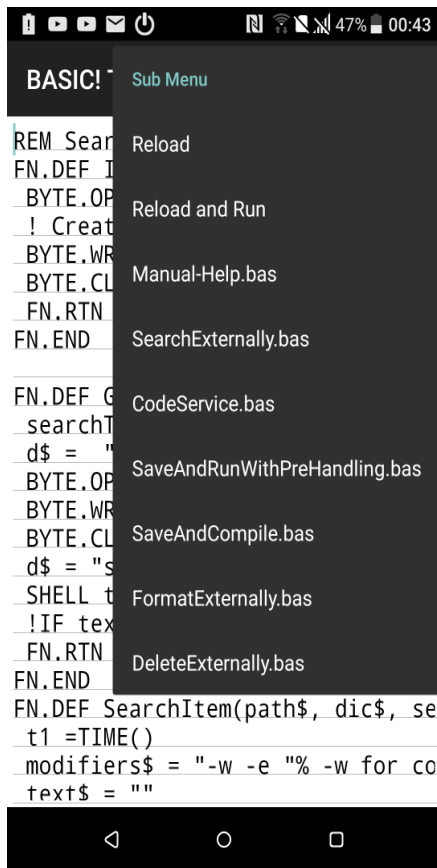
<b>_ShowNavigationbar</b>	0, 1 or 2 (numeric)	<p>If 1 (default) The Navigation bar will be displayed.</p> <p>If 2 The Navigation bar will be transparent displayed. Min. Lollipop 5.0 (API 21)</p> <p>If 0 The Navigation bar will be hidden to the background. Min. Nougat 7.0 (API 24) Will be switched to option 2 or 1 if the current API level is lower.</p>
<b>_NavigationbarColor</b>	{Alpha,}Red,Green,Blue (comma delimited string) or _{Alpha,}ColorName ({{comma delim.} string) or #{hn}hnhnhn (hex. string)	Min. Lollipop 5.0 (API 21)
<b>_NavigationbarLight</b>	0 or 1 (numeric)	<p>If 0 (default) The Navigation bar background is dark. In this case the <b>bar content</b> will be <b>light</b>.</p> <p>If 1 The Navigation bar background is light. In this case the <b>bar content</b> will be <b>dark</b>. Min. Lollipop 5.0 (API 21)</p>
<b>_Menu</b>	Menu Bundle Pointer	Creates additional menu entries. Behind "Stop" and "Editor" if it is running in the development mode. A successful selection will be returned as a human readable JSON string.

See also: [Console.layout](#), [Select](#)

Example

```
bundle.put c, "_Subtitle", " My Sub Title"
bundle.put c, "_TitleHomeEnabled", 1
bundle.put c, "_TitleBackground", "bgColor.png"
bundle.put c, "_TitleShow", 1
bundle.put c, "_TitleIcon", "cartman.png"
```

```
Console.title "My Console Title", c
DO
  PAUSE 50 % Saves power consumption!
UNTIL 0
```



```
! Example: How to Create a Menu
FN.DEF setMenuLayoutBundle()
! "Stop" default in development mode
! "Editor" default in development mode
BUNDLE.PUT item10, "_Title", "Open"
BUNDLE.PB menuLayout, "10", item10
BUNDLE.PUT item30, "_Title", "Save"
BUNDLE.PB menuLayout, "30", item30
BUNDLE.PUT item40, "_Title", "Edit"
BUNDLE.PB menuLayout, "40", item40
BUNDLE.PUT item100, "_Title", "Menu No. 4"
BUNDLE.PUT item100, "_SubMenuStart", 1
BUNDLE.PB menuLayout, "100", item100
BUNDLE.PUT item110, "_Title", "Sub Menu No. 1"
BUNDLE.PUT item110, "_GroupId", 1
BUNDLE.PUT item110, "_Checked", 1
BUNDLE.PB menuLayout, "110", item110
BUNDLE.PUT item120, "_Title", "Sub Menu No. 2"
! Next Line is important! Has to be placed behind the last SubMenu item.
BUNDLE.PUT item120, "_GroupCheckable", 1
! Next Line is important! Has to be placed behind the last SubMenu item.
BUNDLE.PUT item120, "_GroupExclusive", 1
BUNDLE.PUT item120, "_GroupId", 1
BUNDLE.PB menuLayout, "120", item120
BUNDLE.PUT item200, "_Title", "Checker"
! Next Line is important!
! Has to be placed at the next entry behind the last SubMenu item.
BUNDLE.PUT item200, "_AfterSubMenuEnd", 1
BUNDLE.PUT item200, "_Checkable", 1
BUNDLE.PUT item200, "_Checked", 1
BUNDLE.PB menuLayout, "200", item200
BUNDLE.PUT item210, "_Title", "Cartman"
BUNDLE.PUT item210, "_Icon", "cartman.png"
BUNDLE.PB menuLayout, "210", item210
BUNDLE.PUT item220, "_Title", "Galaxy"
BUNDLE.PUT item220, "_Icon", "galaxy.png"
! BUNDLE.PUT item220, "_Enable", 0 % ??? Does still not work.
BUNDLE.PB menuLayout, "220", item220
BUNDLE.PUT item230, "_Title", "Exit"
BUNDLE.PB menuLayout, "230", item230
FN.RTN menuLayout
FN.END
BUNDLE.PB layoutBundle, "_Menu", setMenuLayoutBundle()
```

**Keep in mind, that only one sub menu level is possible on Android.**

```
OnMenuItem:
PRINT "From a Menu"
MENUITEM.GET.DATALINK data$
PRINT data$
! TONE 600, 200
IF IS_IN("Exit", data$) & IS_IN("_Console", data$) THEN END
IF IS_IN("Exit", data$) & IS_IN("_HTML", data$) THEN HTML.CLOSE : myCloser = 1
IF IS_IN("Exit", data$) & IS_IN("_Graphic", data$) THEN EXIT % GR.CLOSE : myCloser = 1
MENUITEM.RESUME
```

### **Console.save <filename\_sexp>**

The current content of the Console is saved to the text file specified by the filename string expression. If the graphics or HTML mode is enabled an error accrues and asks for closing this mode.

See also IS\_GR, IS\_HTML

### **Console.screenshot <filename\_sexp> {,<quality\_nexp>}**

Saves the current screen to a file. The default path is "<pref base drive>/rfo-basic/data/".

The file will be saved as a JPEG file if the filename ends in ".jpg".

The file will be saved as a PNG file if the filename ends in anything else (including ".png").

The optional <quality\_nexp> is used to specify the quality of a saved JPEG file. The value may range from 0 (bad) to 100 (very good). The default value is 50. The quality parameter has no effect on PNG files which are always saved at the highest quality level.

**Note:** The size of the JPEG file depends on the quality. Lower quality values produce smaller files.

### **Screen.rotation, size[], realsize[], density**

Returns inform.....

**It is a description fault!**

The command is not implemented and returns a Syntax Error.

Use instead Screen.size and Screen.rotation.

Example:

```
SCREEN.SIZE size[], realsize[], density
SCREEN.ROTATION rotation
screenRatio = realsize[1]/realsize[2]
IF screenRatio > 1
  IF MOD(rotation, 2) % rotation = 1 | rotation = 3
    BUNDLE.PUT globals, "landscapeRot", 1
    BUNDLE.PUT globals, "portraitRot", 0
  ELSE % rotation = 0 | rotation = 2
    BUNDLE.PUT globals, "landscapeRot", 0
    BUNDLE.PUT globals, "portraitRot", 1
  ENDIF
ELSE
  IF MOD(rotation, 2) % rotation = 1 | rotation = 3
    BUNDLE.PUT globals, "landscapeRot", 0
    BUNDLE.PUT globals, "portraitRot", 1
  ELSE % rotation = 0 | rotation = 2
    BUNDLE.PUT globals, "landscapeRot", 1
    BUNDLE.PUT globals, "portraitRot", 0
  ENDIF
ENDIF
debug.on
debug.dump.bundle globals
```

## JSON

JSON ("JavaScript Object Notation") defines a lightweight data format in which information such as objects, arrays, and other variables can be stored in readable form. In OliBasic, JSON strings can be transformed as Bundles corresponding key strings or converted to XML strings.

JSON is often used to easily exchange information between client and server and is a handy alternative to XML.

See the documentation at [https://www.w3schools.com/js/js\\_json\\_intro.asp](https://www.w3schools.com/js/js_json_intro.asp).

A JSON object begins with a "{" and ends with a "}".

Workaround for a known bundle put issue:

```
GRABFILE j$, "project.json"
```

```
j$ = REPLACE$(j$, "[]", "[" + "\"" + "\"") % Fill empty Arrays with a "" member
```

```
! Now fill empty Strings with n.a.m. = "not a member"
```

```
j$ = REPLACE$(j$, "\"" + "\"", "\"" + "n.a.m." + "\"")
```

### Is\_Json (<json\_sexp>)

Checks if <json\_sexp> is an acceptable or better well done JSON string.

Returns 1 if true otherwise 0 if false.

See also JsonToXml\$(), Bundle.GJ

### XmlToJson\$ (<xml\_sexp>{, <space\_nexp>})

Retruns a JSON string converted from a XML string.

Needs an acceptable or better well done JSON string.

If the optional <spaces\_nexp> returns a number > -1, a structural printout is delivered.

The number of spaces defines the tabulator distance.

If <spaces\_nexp> is -1 or not given, a compact printout is returned.

There may be some post-processing needed, especially if JSON objects beginning with a "{" and ending with a "}" are involved.

If an error occurred, an empty string will be returned.

See also JsonToXml\$(), Bundle.PJ

## HTML Example

```
<!DOCTYPE html>
<html>
<body>

<h2>Store and retrieve data from local storage.</h2>

<p id="JsonOut"></p>
<p id="lastNameOut"></p>
<p id="postalCodeOut"></p>

<script>
var myObj, myJSON, text, obj, obj2;
//Storing data:
myObj = {
  "firstName": "John",
  "lastName": "Smith",
  "age": 25,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": 10021
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "fax",
      "number": "646 555-4567"
    }
  ]
};
myJSON = JSON.stringify(myObj);
localStorage.setItem("testJSON", myJSON);
document.getElementById("JsonOut").innerHTML = myJSON;
// Retrieving data:
text = localStorage.getItem("testJSON");
// First level
obj = JSON.parse(text);
document.getElementById("lastNameOut").innerHTML = obj.lastName;
// Second level
obj2 = obj.address;
document.getElementById("postalCodeOut").innerHTML = obj2.postalCode;
</script>

</body>
</html>
```

## OliBasic Example

! Stringified JSON object

```
myJSON$ = "{\"firstName\": \"John\", \"lastName\": \"Smith\", \"age\": 25, \"address\": {\"streetAddress\": \"21 2nd Street\", \"city\": \"New York\", \"state\": \"NY\", \"postalCode\": 10021}, \"phoneNumbers\": \"[{\"type\": \"home\", \"number\": \"212 555-1234\"}, {\"type\": \"fax\", \"number\": \"646 555-4567\"}] }\""}
PRINT "JsonOut:", myJSON$
```

```
PRINT "Is_json", IS_JSON( myJSON$)
```

! First level

! Workaround for a known bundle put issue:

! Fill **empty Arrays** with a "" member

```
myJSON$ = REPLACE$(myJSON$, "[]", "[" + "\"" + "\"" + "]")
```

! Now fill **empty Strings** with n.a.m. = "not a member"

```
myJSON$ = REPLACE$(myJSON$, "\"" + "\"", "\"" + "n.a.m." + "\"")
```

```
BUNDLE.PJ testJSON, myJSON$
```

```
BUNDLE.GET testJSON, "lastName", lastName$
```

```
PRINT "LastNameOut:", lastName$
```

```
BUNDLE.GET testJSON, "address", obj_address$
```

! Second level

```
PRINT "Is_json", IS_JSON( obj_address$)
```

```
BUNDLE.PJ obj_address, obj_address$
```

```
BUNDLE.GET obj_address, "postalCode", postalCode$
```

```
PRINT "PostalCodeOut:", postalCode$
```

## XML

XML (Extensible Markup Language) is a text-based format for the exchange of structured information. These can be documents, configurations, books, invoices, and more. OliBasic supports XML not in all aspects and is limited by possibilities of the JSON-XML and the XML-JSON parser. So only well formed expressions are parsed. For a well formed output add

`<?xml version="1.0" encoding="utf-8"?>` at the begin of the string.

Unfortunately is an additional DOCTYPE declaration (DTD) entry for a valid XML document needed. See the documentation at <https://www.w3schools.com/xml/default.asp>.

### Is\_Xml (<xml\_sexp>)

Checks if <Xml\_sexp> is an acceptable or better well formed XML string.

Returns 1 if true otherwise 0 if false.

See also `XmlToJson$()`

### JsonToXml\$ (<json\_sexp>{{,<spaces\_nexp>}, <shrink\_nexp>})

Retruns a XML string converted from a JSON string.

Needs an acceptable or better well formed XML string.

If the optional <spaces\_nexp> has a number > -1, a structural printout is delivered.

The number of spaces defines the tabulator distance.

If <spaces\_nexp> is -1 or not given, a compact printout is returned.

If <shrink\_nexp> greater than 0 <spaces\_nexp> will be overwritten by 0 and the line feeds will be removed.

If an error occurred, an empty string will be returned.

Note, that

```
<example name:"Marc" age:31 male:true />
```

returns

```
<example><name>Marc</name><age>31</age><male>true</male></example>
```

if the optional <shrink\_lexp> is switched to off by 0.

See also `XmlToJson$()`

## Bundles

A Bundle is a group of values collected together into a single object. A bundle object may contain any number of string, numeric values **and also there arrays, lists, stacks, bundles, booleans, drawables and bitmaps**. There is no fixed limit on the size or number of bundles. You are limited only by the memory of your device.

The values are set and accessed by keys. A key is string that identifies the value. For example, a bundle might contain a person's first name and last name. The keys for accessing those name strings could be "first\_name" and "last\_name". An age numeric value could also be placed in the Bundle using an "age" key.

A new, empty bundle is created by using the **Bundle.create** command. The command returns a pointer to the empty bundle. Because the bundle is represented by a pointer, bundles can be placed in lists and arrays. Bundles can also be contained in other bundles. This means that the combination of lists and bundles can be used to create arbitrarily complex data structures.

After a bundle is created, keys and values can be added to the bundle using the **Bundle.put** command. Those values can be retrieved using the keys in the **Bundle.get** command. There are other bundle commands to facilitate the use of bundles.

### **Bundle.contain** <pointer\_nexp>{, <key\_sexp> , <contains\_nvar>} ...

If the key specified in the key string expression is contained in the bundle's keys then the "contains" numeric variable will be returned with a non-zero value. The value returned will be zero if the key is not in the bundle. If the bundle does not exist, a new one may be created.

### **Bundle.copy** <SourcePointer\_nexp>, <DestinationPointer\_nexp>

Copies the source bundle to the destination Bundle. The **background** references are cut off!

### **Bundle.create** <pointer\_nvar>

A new, empty bundle is created. The bundle pointer is returned in <pointer\_nvar>.

Example:

```
BUNDLE.CREATE bptr
```

### **Bundle.type** <pointer\_nexp>, <key\_sexp>, <type\_svar>

Returns the value type (string or numeric) of the specified key in the specified string variable. The <type\_svar> will contain an uppercase "N" if the type is numeric. The <type\_svar> will contain an uppercase "S" if the type is a string. **For arrays the dimensions are added like S[6] or N[2,2,5,..]. The other types result in "List", "Stack or Object" and "Bundle". If the bundle does not exist or does not contain the requested key, the command generates a run-time error. If you get "Stack or Object" and you are not sure, dump the Bundle with BUNDLE.DUMP.BUNDLE first. In doubt use the type Object.**

Example:

```
BUNDLE.TYPE bptr, "age", type$  
PRINT type$ % will print N, see also Bundle.put
```



**Bundle.put** <pointer\_nexp>, <key\_sexp>, <value\_nexp>|Array[]|<value\_sexp>|Array\$[]{ ..., <key\_sexp>, <value\_nexp>|Array[]|<value\_sexp>|Array\$[] ...}

The value expression or array will be placed into the specified bundle using the specified key. If the bundle does not exist, a new one may be created.

The type of the value will be determined by the type of the value expression.

Example:

```
BUNDLE.PUT bptr, "first_name", "Frank", "last_name", "Musterman"  
BUNDLE.PUT bptr, "age", 44
```

**Bundle.PL** <pointer\_nexp>, <key\_sexp>, <list\_ptr\_nexp>

Puts a list into a bundle. Read Bundle.Put.List instead Bundle.PL

The list will be placed into the specified bundle using the specified key. If the bundle does not exist, a new one may be created.

The type of the value is a list pointer.

Example:

```
BUNDLE.PL bptr, "names", llistPtr
```

**Bundle.PS** <pointer\_nexp>, <key\_sexp>, <stack\_ptr\_nexp>

Puts a stack into a bundle. Read Bundle.Put.Stack instead Bundle.PS

The stack will be placed into the specified bundle using the specified key. If the bundle does not exist, a new one may be created.

The type of the value is a stack pointer.

Example:

```
BUNDLE.PL bptr, "toDos", stackPtr
```

**Bundle.PV** <pointer\_nexp>, <key\_sexp>, <boolean\_nexp|Array[]>

Puts a **boolean** value or array into a bundle. Read Bundle.Put.Verum instead Bundle.PV  
**Verum** is Latin for Truth.

The **boolean** will be placed into the specified bundle using the specified key. If the bundle does not exist, a new one may be created.

The type of the **boolean** value is a numeric expression or array. If <boolean\_nexp> or an entry of an array is > 0 the bundle entry will be saved as **true**. Otherwise **false** is saved. The background references are cut off!

Example:

```
BUNDLE.PV bptr, "globals", 1
```

**Bundle.PB** <pointer\_nexp>, <key\_sexp>, <bundle\_pointer\_nexp>

Puts a bundle into a bundle. Read Bundle.Put.Bundle instead Bundle.PB

The bundle <bundle\_pointer\_nexp> will be placed into with <pointer\_nexp> specified bundle using the specified key. If the bundle specified <pointer\_nexp> does not exist, a new one may be created.

The type of the value is a bundle pointer. The background references are cut off!

Example:

```
BUNDLE.PL bptr, "globals", bundlePtr
```

### Bundle.PJ <pointer\_nexp>, <json\_sexp>

Puts the content of a JSON string into a bundle. Read Bundle.Put.JSON instead Bundle.PJ

The first level of the JSON content is parsed to bundle-supported types.

Arrays of JSON objects will be stored as Arrays of strings and have to be parsed outside the bundle. Which should be also parsed as bundles.

Example:

```
jsonString$ = XmlToJson$ ("<name>Nicolas</name>") % Native XML string
! Workaround for a known bundle put issue:
! Fill empty Arrays with a "" member
jsonString$ = REPLACE$(jsonString$, "[ ]", "[" + "\"" + "\"" + "]")
! Now fill empty Strings with n.a.m. = "not a member"
jsonString$ = REPLACE$(jsonString$, "\"" + "\"", "\"" + "n.a.m." + "\"")
BUNDLE.PJ bptr, jsonString$
BUNDLE.PJ bptr, "{\"age\":38, \"city\": \"Paris\"}" % Native JSON string
BUNDLE.PV bptr, "male", 1 % Boolean is true
preJSON$ = " 'street' : 'Main Street' , 'no' : 12 , 'FR' : false "
BUNDLE.PJ bptr, "{" + Replace$(preJSON$, "", chr$(34)) + "}" % chr$(34) equals "\""
BUNDLE.GJ bptr, newJson$, -1 % Returns a compact JSON string
PRINT newJson$ % Prints the result
```

```
"{"street": "Main Street", "US": "true", "no": 12, "age": 38, "city": "Paris", "male": true, "name": "Nicolas"}"
```

See also XmlToJson\$(), Is\_Json(), Is\_Xml()

### Bundle.PP <pointer\_nexp>, <key\_sexp>, <bitmap\_pointer\_nexp> **Deprecated**

#### Gr.bitmap.put <bundle\_pointer\_nexp>, <key\_sexp>, <bitmap\_pointer\_nexp>

Puts a bitmap into a bundle.

The bitmap will be placed into the specified bundle using the specified key. If the bundle does not exist, a new one may be created.

The type of the value is a bundle pointer. The **background** references are cut off!

Needs Graphic Mode.

Example:

```
Gr.bitmap.put bptr, "picture1", bitmapPtr
```

#### Gr.drawable.put <bundle\_pointer\_nexp>, <key\_sexp>, <drawable\_pointer\_nexp>

Puts a drawable **as a bitmap** into a bundle.

The drawable will be placed into the specified bundle using the specified key. If the bundle does not exist, a new one may be created.

The type of the value is a bundle pointer. The background references are cut off!

Needs Graphic Mode.

Example:

```
Gr.drawable.put bptr, "picture1", drawablePtr
```

### Bundle.get <pointer\_nexp>, <key\_sexp>, <value\_nexp>|Array[]|<value\_sexp>|Array\$[]|{ ..., <key\_sexp>, <value\_nexp>|Array[]|<value\_sexp>|Array\$[] ...}

Places the value specified by the key string expression into the specified numeric or string variable. The type (array, string or numeric) of the destination variable must match the type stored with the key. **The exception is an Object as an incoming data type. This Object will**

be converted as much as possible into a string value. If the bundle does not exist or does not contain the requested key, the command generates a run-time error.

If a number is stored as a String, Bundle.get is in some cases also able to return the value as a number.

Example:

```
BUNDLE.GET bptr, "first_name", first_name$, "last_name", last_name$  
BUNDLE.GET bptr, "age", age  
BUNDLE.GET bptr, "professions", professions$[]
```

### Bundle.GL <pointer\_nexp>, <key\_sexp>, <list\_ptr\_nexp>

Read Bundle.Get.List instead Bundle.GL

Places the list specified by the key string expression into the pointer specified list. The type (string or numeric) of the destination list must match the type stored with the key. If the bundle does not exist or does not contain the requested key, the command generates a run-time error.

Example:

```
BUNDLE.GL bptr,"names", listPtr
```

### Bundle.GS <pointer\_nexp>, <key\_sexp>, <stack\_ptr\_nexp>

Read Bundle.Get.Stack instead Bundle.GS

Places the stack specified by the key string expression into the pointer specified stack. The type (string or numeric) of the destination stack must match the type stored with the key. If the bundle does not exist or does not contain the requested key, the command generates a run-time error.

Example:

```
BUNDLE.GS bptr,"toDos", stackPtr
```

### Bundle.GV <pointer\_nexp>, <key\_sexp>, <boolean\_nval|Array[]>

Read Bundle.Get.Verum instead Bundle.GV. Verum is Latin for Truth.

Places the **boolean** specified by the key string expression into with <boolean\_nval> specified numeric variable which can be also an array. If the **boolean** bundle content is **true** it returns 1. Otherwise (**false**) 0 is returned. If the bundle does not exist or does not contain the requested key, the command generates a run-time error.

Example:

```
BUNDLE.GV bptr,"boolean", myBoolean
```

### Bundle.GB <pointer\_nexp>, <key\_sexp>, <bundle\_ptr\_nvar>

Read Bundle.Get.Bundle instead Bundle.GB

Places the bundle specified by the key string expression into the pointer <bundle\_ptr\_nvar> specified bundle. If the bundle specified by <pointer\_nexp> does not exist or does not contain the requested key, the command generates a run-time error. If the bundle specified by <bundle\_ptr\_nvar> does not exist a new one will be created.

Example:

```
BUNDLE.GB bptr,"bundleContainer", bundlePtr
```

### **Bundle.GJ <pointer\_nexp>, <json\_sexp>{, <spaces\_nexp>}**

Gets the content of a bundle into a JSON string. Read `Bundle.Get.JSON` instead `Bundle.GJ`.

Supported data types are Strings, Doubles, Booleans as single value or Array.

If the optional <spaces\_nexp> returns a number > -1, a structural printout is delivered.

The number of spaces defines the tabulator distance.

If <spaces\_nexp> is -1 or not given, a compact printout is returned.

There may be some post-processing needed, especially if JSON objects beginning with a "{" and ending with a "}" are involved.

Note that multi-dimensional arrays are converted into **one-dimensional** arrays!

Example:

```
BUNDLE.GJ bptr, jsonString$  
Byte.open w, ftb, path$+fileName$  
Byte.write.buffer ftb, jsonString$  
Byte.close ftb
```

See also `JsonToXml$()`, `Is_Json()`

**Bundle.GP** <pointer\_nexp>, <key\_sexp>, <bitmap\_pointer\_nexp> **Deprecated**

**Gr.bitmap.get** <bundle\_pointer\_nexp>, <key\_sexp>, <bitmap\_ptr\_nexp>

Gets a bitmap from a bundle.

Places the bitmap specified by the key string expression into the pointer specified bitmap.

If the bundle does not exist or does not contain the requested key, the command generates a run-time error.

Needs Graphic Mode.

Example:

```
Gr.bitmap.get bptr,"Picture2", bitmapPtr
```

**Gr.drawable.get** <bundle\_pointer\_nexp>, <key\_sexp>, <drawable\_ptr\_nexp>

Gets a drawable **in the form of a bitmap** from a bundle.

Places the drawable specified by the key string expression into the pointer specified drawable. If the bundle does not exist or does not contain the requested key, the command generates a run-time error.

Needs Graphic Mode.

Example:

```
Gr.drawable.get bptr,"Picture2", drawablePtr
```

**Bundle.in** <recAction\_sexp>, <retData\_svar>, <retBundleIndex\_nvar>

Receives an Intent from a calling app or launcher. The parameter recAction returns the calling action. With retData you get the Data URI string from the Intent. The retBundleIndex returns a bundle. If no data is broadcasting retData returns "" and retBundleIndex returns an empty Bundle. Use DECODE\$ with the type "URL" and the Qualifier "charset" like DECODE\$ ("URL", "charset", retData\$) if necessary.

Parameters in the retData String have to be in URI style, only.

The returned Bundle with the retBundleIndex pointer contains the received Extras.

**GitHub#174**

**Bundle.out** <sendAction\_sexp>, <sendData\_sexp>, <sendExtraBundle\_pointer\_nexp>

Sends a result as an Intent return to a calling app or launcher, when the program is finished. The App or BASIC! has to be called **inclusive component name** (normally package name + ".Basic"). If you launch the program in the Editor, you get normally **no result**.

The parameter sendAction sends an action (extremely seldom) back. With sendData you send a Data Extra string (seldom) back. Android accept only URIs! The retBundleIndex sends a bundle back.

The sent Bundle with the sendExtraBundleIndex pointer contains the sent Extras.

In this case we have two bundle levels:

First level:

Supporting **only** the Intent Extras Types **Double, String and Bundle**.

**Other types are automatically removed.**

Second level:

If you use on the first level a Bundle that contains also one or more Bundles, you can use these as full featured BASIC! Bundles.

### **Bundle.save <pointer\_nexp>, <fileName\_sexp>**

Saves a bundle into a file. Only recommended for temporary use, because the internal coding of the result is operating system version depended. It seems, that newer Android versions are able to read older bundle files. This is important, if the user want to upgrade his operating system. But this behavior is without any guarantee.

Bitmaps in bundles are also not supported in this case.

Maybe there is a size limitation. Not tested yet.

Example:

```
BUNDLE.SAVE bptr, "saveState.bun"
```

See also [Bundle.PJ](#), [Byte.write.buffer](#), [Grabfile](#), [Bundle.GJ](#)

### **Bundle.load <pointer\_nexp>, <fileName\_sexp>**

Loads a bundle from a file. Only recommended for temporary use, because the internal coding is operating system version depended.

Bitmaps in bundles are also not supported in this case.

Maybe there is a size limitation. Not tested yet.

Example:

```
BUNDLE.LOAD bptr, "saveState.bun"
```

### **Bundle.remove <pointer\_nexp>, <key\_sexp>**

Removes the key named by the string expression <key\_sexp>, along with the associated value, from the bundle pointed to by the numeric expression <pointer\_nexp>. If the bundle does not contain the key, nothing happens. If the bundle does not exist, a new one may be created.

### **Bundle.clear <pointer\_nexp>**

The bundle pointed to by <pointer\_nexp> will be cleared of all tags. It will become an empty bundle. If the bundle does not exist, a new one may be created.

### **Bundle.kill.last**

Kills the last Bundle of the internal Bundles list. Bundles are global. If you create a Bundle within a function so you are able to kill this Bundle before leaving the function.

### **Debug.dump.bundle <bundlePtr\_nexp>**

Dumps the Bundle pointed to by the Bundle Pointer numeric expression.

If a Bundle contains other types as single numeric values from type Double or String, this value will be converted and printed as a string without quotation marks.

**Email.send** <recipient\_sexp>|Array\$[], <subject\_sexp>, <body\_sexp>~  
 {{{,<sendTo\_sexp>},<cC\_sexp>|Array\$[],<bCC\_sexp>|Array\$[],~  
 <attachment\_svar>|Array\$[]}

The email message in the Body string expression will be sent to the named recipient(s) with the named subject heading.

<sendTo\_sexp> specifies an app which should send. (Ex.: Gmail, WhatsApp, DropBox or OneDrive. See also App.installed.) <cC\_sexp> and <bCC\_sexp> place the CC and BCC recipient(s) in the mail header. <attachment\_svar> specifies the attachment(s). For a single file only a single String variable is accepted.

The command expects double quotes for undefined arguments.

Files from the \_Internal file folder have to be copied into an external folder before sending. In case of Android 10+ with Scoped Storage you have to copy your files into the File Provider directory.

! You can get the File Provider directory path by

```
File.root fpPath$, "_FileProvider".
```

! Copy your files

```
File.copy "whee.mp3", fpPath$ + "/" + "whee.mp3"
```

! Now set the permissions for reading.

```
Bundle.put pB, "_FileP_Read", 1
```

```
Provider pB
```

```
...
```

```
Program.info blnf
```

```
Bundle.get blnf, "_PackageName", pn$
```

```
fpFromOutside$ = "content://" + pn$ + "/" + "whee.mp3" % Path within the File Provider dir.
```

```
Emal.send ... , fpFromOutside$
```

```
...
```

! Withdraw the authorization

```
Bundle.put pB, "_FileP_Read", 0
```

```
Provider pB
```

! Usually delete the copied files after sending

```
File.delete fpPath$ + "/" + "whee.mp3"
```

See also Provider, File.copy, File.Move, File.Delete

Example:

```
gmail$ = "com.google.android.gm"
```

```
Array.load cC$[], "bob.ex@example.com", "harry.c@examples.com"
```

```
FILE.ROOT dataPath$
```

```
Array.load files$[], "file://" + dataPath$ + "/" + "cartman.png",~
```

```
"file://" + dataPath$ + "/" + "whee.mp3"
```

```
Email.send "b.ex@gmail.com", "Mail Subject", "Message", gmail$, cC$[], "", files$[]
```

GitHub#153

### Pause <ticks\_nexp>

Stops the execution of the BASIC! program for <ticks\_nexp> milliseconds. One millisecond = 1/1000 of a second. Pause 1000 will pause the program for one second. **A pause can not be interrupted.**

An infinite loop can be a very useful construct in your programs. For example, you may use it to wait for the user to tap a control on the screen. A tight spin loop keeps BASIC! very busy doing nothing. A **Pause**, even a short one, reduces the load on the CPU and the drain on the battery. Depending on your application, you may want to add a **Pause** to the loop to conserve battery power:

```
DO : PAUSE 50 : UNTIL x <> 0
```

### FOR <nvar>, Array[]|Array\$[] / Next

Initiates a FOR/NEXT loop. The index given by <nvar> starts with 1 and counts up until the total length of an array given by Array[]|Array\$[]. If the array has more than one dimension, the index counts up column by column.



## Interrupt Labels (Event Handlers)

You can perform physical actions that tell your BASIC! program to do something. When you touch the screen or press a key you cause an *event*. These events are *asynchronous*, that is, they happen at times your program cannot predict. BASIC! detects some events so your program can respond to them.

BASIC! handles events as *interrupts*. Each event that BASIC! recognizes has a unique *Interrupt Label*. When an event occurs, BASIC! looks for the Interrupt Label that matches the event.

- If you have not written that Interrupt Label into your program, the event is ignored and your program goes on running as if nothing happened.
- If you have included the right Interrupt Label for the event, BASIC! jumps to that label and continues execution at the line after the label. This is called trapping the event.

BASIC! does not necessarily respond to the event as soon as it occurs. The statement that is executing when the event occurs is allowed to complete, then BASIC! jumps to the Interrupt Label.

When you use an Interrupt Label to trap an event, BASIC! executes instructions until it finds a *Resume* command that matches the Interrupt Label. During that time, it records other events but it does not respond to them. The block of code between the Interrupt Label and the matching Resume may be called an *Interrupt Service Routine (ISR)* or, if you prefer, an *Event Handler*.

**In this version variables in functions are per default local if an interrupts traps! If you get in trouble with older code, use in addition GLOBASL.ALL after the colon (on....:) and GLOBALS.NONE before RESUME.**

When BASIC! executes the event's Resume command, it resumes normal execution.

- BASIC! jumps back to where it was running when the interrupt occurred.
- BASIC! again responds to other events, including any that occurred while it was handling an event. **\*\*\* In other words events are be put in a waiting queue. Thus many events can be triggered and will be handled on the first in first out principle, but OnError: is always serviced first.**

For the interrupt handling to be as fast as possible:

- Use a timer interrupt instead of pause in your main loop, that solves a lot of interrupt issues, because PAUSE also delays the interrupts.
- In this version you can use Sched.set as second timer, too.

**\*\*\* Need more investigation**

An Interrupt Label looks and behaves just like any other label in BASIC!. However, you must not execute any of the Resume commands except to finish an event's handler.

## All Interrupt Labels

BASIC! supports trapping of the following events. These interrupt labels and their Resume commands are described in various parts of this manual.

- **OnBackground:**            **Background.resume**
- **OnBackKey:**            **Back.resume**
- **OnBroadcast:**           **Broadcast.resume**
- **OnBtReadReady:**       **Bt.onReadReady.resume**
- **OnBtStatus:**           **Bt.onStatus.resume**

- **OnConsoleTouch:** **ConsoleTouch.resume**
- **OnError:** None (not a true event, see **OnError:**, below)
- **OnGrTouch:** **Gr.onGrTouch.resume**
- **OnGrTouchMove:** **Gr.onGrTouchMove.resume**
- **OnGrTouchUp:** **Gr.onGrTouchUp.resume**
- **OnHtmlReturn:** **Html.onHtmlReturn.resume**
- **OnGrScreen:** **Gr.onGrScreen.resume**
- **OnKbChange:** **Kb.resume**
- **OnKeyDown:** **KeyDown.resume**
- **OnKeyPress:** **Key.resume**
- **OnLowMemory:** **LowMemory.resume**
- **OnMenuKey:** **MenuKey.resume**
- **OnMenuItem:** **MenuItem.resume**
- **OnSched:** **Sched.resume**
- **OnTimer:** **Timer.resume**
- **OnUsbReadReady:** **Usb.onReadReady.resume**
- **OnUsbStatus:** **Usb.onStatus.resume**

### OnError:

Special interrupt label that traps a run-time error as if it were an event, except that:

- **OnError:** has no matching Resume command. You can use **GoTo** to jump anywhere in your program.
- **OnError:** is not locked out by other interrupts.
- **OnError:** does not lock out other interrupts.

If a BASIC! program does not have an **OnError:** label and an error occurs while the program is running, an error message is printed to the Output Console and the program stops running.

If the program does have an **OnError:** label, BASIC! does not stop on an error. Instead, it jumps to the **OnError:** label like a **GOTO** instead of a **GOSUB/RETURN** (see "Interrupt Labels", above). The error message is not printed, but it can be retrieved by the **GETERROR\$()** function. If you jumps from a function its variables are still local in its area also.

Be careful. An infinite loop will occur if a run-time error occurs within the **OnError:** code. You should not place an **OnError:** label into your program until the program is fully debugged. Premature use of **OnError:** will make the program difficult to debug.

Example:

```
REM Function Exeption Handler
FN.DEF myFunc()
  func$ = "myFunc"
  p = p + 4 : z = 7 : k = 5
  p$ = p % Line with Error code.
  labelGoOn_myFunc:
  PRINT p, z, k
FN.END
```

```
p = 23
z = 44
k = 99
myFunc()
END
```

```
ONERROR:
IF func$ = "myFunc"
  PRINT GETERROR$()
  ! GLOBALS.FNIMP z % GLOBALS.FNIMP, GLOBALS.ALL and GLOBALS.NONE
  ! together are not possible in the same function
  PRINT p % returns 4, because still in the function myFunc
  PRINT z % returns 44, because imported in the area of the function
  PRINT k % returns 0, because not imported
  GLOBALS.ALL % Usefull for exeption handling, the value of the variable is not imported!
  PRINT k % returns 99, because access to the global symbol table
  GLOBALS.NONE
  GOTO labelGoOn_myFunc
ENDIF
```

### OnKeyDown:

Interrupt label that traps a tap\_down on any key except the volume keys. They have to be switched on with VolKeys.On. BASIC! executes the statements following the

**OnKeyDown:** label until it reaches a **KeyDown.resume**.

### KeyDown.resume

Resumes execution at the point in the BASIC! program where the **OnKeyPress:** interrupt occurred.

### OnKeyPress:

Interrupt label that traps a tap\_up on any key. BASIC! executes the statements following the **OnKeyPress:** label until it reaches a **Key.resume**. (OnKeyUp would describe it better.)

### Key.resume

Resumes execution at the point in the BASIC! program where the **OnKeyPress:** interrupt occurred.

### **OnMenuItem:**

Interrupt label that traps if a menu item is selected. BASIC! executes the statements following the **OnMenuItem:** label until it reaches a **OnMenuItem.resume**.

### **MenuItem.get.datalink <data\_svar>**

Returns the data of the last selected menu item in a human readable JSON string.

```
Begin: "{" +~
      "_MenuFrom:_Graphic" +~ % Other cases are "_Console" and "_HTML"
      "_TitleText:MyItemTitle" +~ % If is "_TitleUp" returned, the Home Button of
                                   % the Title Bar is selected!
      "_ItemId:20" +~           % Returns the item id
      "_GroupId:1" +~          % Returns the group id
      "_Checked:1"+~           % Returns if checked 1 or not 0
      "_Checkable:0"+~         % Returns if it is check-able 1 or not 0;
End:  "}"
```

### **MenuItem.resume**

Resumes execution at the point in the BASIC! program where the **OnMenuItem:** interrupt occurred.

### Locals.on

After this command the variables in functions are definitely **local**.

### Locals.off

After this command the variables in functions can be **global**, if Globals.all is used.

### Globals.all

After this command all variables in functions are **global**.

With Locals.on and Locals.off you are able to change variables in parts or in the entire function to local.

**Be very carefully with this command!** With the **Include** command and some libs like GW.bas you will get in trouble if you do not put the lib calls, **Locals.on** and **Locals.off**, in brackets. **The use inside from functions is strongly not recommended, because the results are wrong if the function is a part of a function call chain inside an interrupt routine.**

A better alternative is to use bundles. They are global, thus a global container with all the variable types is available. Take care if you including a library like GW.bas. It uses the 1 as the global bundle pointer. Thus use at the first line in your main program part **INCLUDE** "GW.bas" and afterwards **BUNDLE.CREATE** myGlobals. In this case myGlobals is 2. Setting the global bundle pointer into **FN.DEF** (myGlobals, ...) makes your code readable. See also Globals.fnimp

### Globals.none

After this command the variables in functions are **local**, **except the variables imported by Globals.fnimp**.

### Globals.fnimp <varexp> {... , <varexp>}, ...

#### **Fn.import <varexp> {... , <varexp>}, ...**

Imports variables from the main program area in a function. Thus you have access to the specified variables from the main area.

Credits to Humpty for this enhancement.

**GLOBALS.FNIMP**, **GLOBALS.ALL** and **GLOBALS.NONE** together are not possible in the same function.

### **GoTo.get.index <nvar>{, <lastChar\_nvar>}**

Returns the current execution command index.

The last character of current execution command will be returned by <lastChar\_nvar>.

### **GoTo.get.error.index <nvar>{, <lastChar\_nvar>}**

Returns the execution command index at the last error.

The last character of current execution command will be returned by <lastChar\_nvar>.

### **GoTo.set.index <nexp>**

Jumps to the given execution command. Note, there is no possibility check.

If <nexp> is greater than the size of the command list, it jumps to the last command!

Example:

```
! : PRINT "ci", ci  counts as an extra command
GOTO.GET.INDEX ci, lastChar : PRINT "ci", ci
pring % A command with trouble
PRINT "OK"
END
ONERROR:
GOTO.GET.ERROR.INDEX eri
PRINT eri
GOTO.SET.INDEX eri + 1 % Go to the command behind (+1) the error
```

Example:

```
! How to get the program line from the above lastChar
! GetProgLineNum.bas
FILE.ROOT path$, "_Source"
FILE.SELECT progPath$, path$
TEXT.OPEN r, ftb, progPath$
counter = 0
mChars = 0
INPUT "Insert character number", lastChar
DO
  counter ++
  TEXT.READLN ftb, line$
  mChars = mChars + LEN(line$) + 1 % + 1, because LF
  TEXT.EOF ftb, mEof
  IF mChars >= lastChar THEN mEof = 1
UNTIL mEof
TEXT.CLOSE ftb
PRINT "Line: "; counter
PRINT line$
```

Example:

```
GOTO.GET.INDEX ci
BUNDLE.PUT gbp, "Try01", ci
FN.DEF myFunc01 (gbp, data)
PRINT "FN 01"
result01 = data * 3/ % Returns an error
FN.RTN result01
BUNDLE.PUT gbp, "NewResult", r
FN.RTN r
FN.END
GOTO.GET.INDEX ci
BUNDLE.PUT gbp, "Catch01", ci - 3

GOTO.GET.INDEX ci
BUNDLE.PUT gbp, "Try02", ci
FN.DEF myFunc02 (gbp, data)
PRINT "FN 02"
result02 = data * "2" % Returns an error
FN.RTN result02
BUNDLE.GET gbp, "NewResult", r
FN.RTN r
FN.END
GOTO.GET.INDEX ci
BUNDLE.PUT gbp, "Catch02", ci - 3
DEBUG.ON
DEBUG.DUMP.BUNDLE gbp
res = myFunc02 (gbp, 33)
PRINT res
res = myFunc01 (gbp, 33)
PRINT res
PRINT "OK"
DO
  PAUSE 100
UNTIL 0
END

ONERROR:
GOTO.GET.ERROR.INDEX eri
PRINT "Error at Execution Command", eri
BUNDLE.GET gbp, "Try02", Try02
BUNDLE.GET gbp, "Catch02", Catch02
IF eri > Try02 & eri < Catch02 THEN ~
: BUNDLE.PUT gbp, "NewResult", 44 : GOTO.SET.INDEX Catch02
PRINT GETERROR$()
END
```

Instead of a Bundle you can also use Globals.fnimp, Fn.import, Tries[], Catches[] etc.

## Broadcasts

Broadcasts are messages from the system or other applications (activities). You can send and receive these messages with broadcasts. Before you are able to receive broadcasts, you have to initialize a so called Broadcast Receiver. The Broadcast Message is called an Intent. The Intent will be received by any application that has the right Intent Filter. If you send messages between instances of BASIC! (different package IDs are needed) at runtime, maybe use this action addresses:

- Prog 1 use for sending "com.rfo.basicFellow.broadcast.SEND1" and receiving "com.rfo.basicOli.broadcast.SEND2"
- Prog 2 use for sending "com.rfo.basicOli.broadcast.SEND2" and receiving "com.rfo.basicFellow.broadcast.SEND1"

The use of APP.Broadcast is not recommended, but APP.SAR can be used to send a Broadcast with much more options.

Example:

```
LIST.CREATE S, commandListPointer
LIST.ADD commandListPointer~
"new Intent("+CHR$(34)+"com.rfo.basic.broadcast.SEND"+CHR$(34)+");" ~
! → );" ~ ← is important
"EOCL"
BUNDLE.PL appVarPointer,"_CommandList",commandListPointer
BUNDLE.Put appVarPointer,"_Broadcast",""
APP.SAR appVarPointer
```

## Broadcast.init <action\_sexp> | Array\$[]

Initializes a Broadcast Receiver

## OnBroadcast:

Interrupt label that traps a received broadcast. BASIC! executes the statements following the **OnBroadcast:** label until it reaches a **Broadcast.resume**.

## Broadcast.in <recAction\_sexp>, <retData\_svar>, <retBundleIndex\_nvar>

Receives a Broadcast message (Intent) according to the recAction action adress. With retData you get the Data Extra string from the Intent. The retBundleIndex returns a bundle. If no data is broadcasting retData returns "" and retBundleIndex returns an empty Bundle. You have also the option to detect system broadcasts, but in some cases you need also the right permissions. As an example you need for the broadcast filter "android.net.conn.INET\_CONDITION\_ACTION" the new permission ACCESS\_NETWORK\_STATE. Today the two available compilers are still not capable for broadcast permission autodetection. That is your job. A compressed information you can find under: <https://android.googlesource.com/platform/frameworks/base/+master/core/res/AndroidManifest.xml>

## Broadcast.resume

Resumes execution at the point in the BASIC! program where the **OnBroadcast:** interrupt occurred.



## Broadcast.close

Closes the Broadcast Receiver

## Broadcast.bundle <sndAction\_sexp>, <key\_sexp>, <bundle\_ptr\_nvar>{, <ordered\_nexp>}

Sends a Broadcast message as a Bundle with the action address sndAction and the key. An ordered Broadcast is send, if <ordered\_nexp> is > 0. Default is 0.

## Broadcast.string <sndAction\_sexp>, <key\_sexp>, <msg\_sexp>{, <ordered\_nexp>}

Sends a Broadcast message as a String with the action address sndAction and the key. An ordered Broadcast is send, if <ordered\_nexp> is > 0. Default is 0.

## KB.send.keyevent <tapType\_nexp>, <actionType\_sexp>, <keyCode\_nexp>

Sends a key event internally as a Broadcast message to the environment.

Supported tap types <tapType_nexp>	
0	Key down
1	Key up
4	Key down and up

Action type group <actionType_sexp>	
Value	Intent action (informative)
_Global (no permission)	android.intent.action.GLOBAL_BUTTON
_Call	android.intent.action.CALL_BUTTON
_Media	android.intent.action.MEDIA_BUTTON
_Camera	android.intent.action.CAMERA_BUTTON

For key events <keyCode\_nexp> see:

<https://android.googlesource.com/platform/frameworks/base/+/master/core/java/android/view/KeyEvent.java>

Mostly apps ignore these events. Try some media player like VLC, Gplayer or TotalCommander.

Example:

```
upAndDown = 4
type$ = "_media"
keyCode = 127 %Media PAUSE button
kb.send.keyevent upAndDown, type$, keyCode
pause 3000
kb.send.keyevent upAndDown, type$, keyCode
```

### Inkey\$ <svar>{{, <rawKeyEvent\_svar>}, <utf-8\_svar>}

Reports key taps for the a-z, 0-9, Space and the D-Pad keys. The key value is returned in <svar>.

The D-Pad keys are reported as "up", "down", "left", "right" and "go". If any key other than those have been tapped, the string "key nn" will be returned. Where nn will be the Android key code for that key.

If no key has been tapped, the "@" character is returned in <svar>.

Keep in mind, that **soft** keyboards send a limited character set. Characters like "°♠♥♦♣ 《》 ¡¿äöü" are only supported by **USB** or **Bluetooth** keyboards or other input devices like game pads in the case of this function.

Rapid key taps are buffered in case they come faster than the BASIC! program can process.

With rawKeyEvent you get an optional raw key event description with action, keyCode, scanCode, metaState, flags, repeatCount, eventTime, downTime, deviceId and source values.

As an option you get with <utf-8\_svar> the UTF-8 character back.

If you want correct results use ONKEY...: interrupt handling instead a DO – UNTIL loop.

But do not use the command PAUSE if have a lot keystrokes in conjunction with key event handling.

Example

```
KEYDOWN.ON % The opposite is KEYDOWN.OFF
DO
UNTIL 0
```

```
ONKEYDOWN: %A Key Is Down interrupt
! The second string, raw key event parameter
INKEY$ mKey$, mKeyEvent$, mUniKeyEvent$
PRINT "Got "; mKey$, " "; mKeyEvent$, mUniKeyEvent$
KEYDOWN.RESUME %Resumes execution at the point BASIC! program where
! the OnKeyDown: interrupt occurred.
```

```
ONKEYPRESS: %Imo ONKEYUP points the fact better
INKEY$ mKey$, mKeyEvent$, mUniKeyEvent$
PRINT "Got "; mKey$, " "; mKeyEvent$, mUniKeyEvent$
KEY.RESUME
```

### Keydown.on

Key Down Detection possible.

### Keydown.off

No Key Down Detection. Default status.

### Gr.set.dashpatheffect {<intervals\_list\_ptr\_nvar>{, <phase\_nexp>}}

Path Pattern:

The intervals list must contain an even number of entries ( $\geq 2$ ), with the even indices specifying the "on" intervals, and the odd indices specifying the "off" intervals.

Phase:

Phase is an offset into the intervals list ([modifies](#) the sum of all of the intervals).

The intervals list controls the length of the dashes.

The GR.set.stroke controls the thickness of the dashes.

Note: This path effect only affects drawing with the GR.color's style parameter is set to 0 (STROKE) or 2 (STROKE\_AND\_FILL). It is ignored if the drawing is done with style == 1 (FILL).

A simple [Gr.set.dashpatheffect](#) without any arguments clears the effect to a full line.

Example:

```
scale = 2
LIST.CREATE N, pathPatternType1
LIST.ADD pathPatternType1, 10*scale, 7*scale, 3*scale, 7*scale
GR.SET.DASHPATHEFFECT pathPatternType1, 5*scale
```

### Gr.path <obj\_nvar>, <list\_pointer\_nvar> {, <x\_nexp>, <y\_nexp>}

Creates a path object defined by a list of strings with the system values "\_MoveTo", "\_LineTo", "\_QuadTo" (Quadratic Bezier Curve), "\_CubicTo" (Cubic Bezier Curve), "[\\_ArcSegTo](#)", "\_ArcTo", "\_Close" and "\_End". The path will be located within the bounds of the parameters. The x and y parameters can be set to move the path. The path will or will not be filled depending upon the **Gr.color** style parameter. The <obj\_nvar> returns the Object List object number for this rectangle. This object will not be visible until the **Gr.render** command is called.

The **Gr.modify** parameters for **Gr.path** are: "x", "y", "list", "paint" and "alpha".

.

Example:

```
GR.OPEN "_White", 0, 1
GR.COLOR "_Blue", 0
LIST.CREATE S, pathDraft1
cPx = 250 : cPy = 250 % Center Points
R4 = 230
Angle1 = 30
Angle2 = 45
Angle3 = 60
! The following coordinates are relative values according to the optional placing values.
LIST.ADD pathDraft1, ~
!"_MoveTo"~ %The first _MoveTo call is like GR.poly included.
STR$(cPx), STR$(cPy+R4)~ %Point 1
!!b1
Set the beginning of the next contour to the point (x1,y1).
Moving like a pen in a pen plotter.
Parameters:
x1    The x-coordinate of the start of a new contour
y1    The y-coordinate of the start of a new contour
!!e1
"_LineTo"~
```

STR\$(cPx-R4\*COS(Angle2\*PI()/180)),STR\$(cPy+R4\*SIN(Angle2\*PI()/180))~ %Point 2  
!!b2

Add a line from the last point {(x1,y1)} to the specified point (x2,y2). If no \_MoveTo call has been made for this contour, the first point is automatically set to (0,0).

Parameters:

x2 The x-coordinate of the end of a line

y2 The y-coordinate of the end of a line

!!e2

"\_LineTo"~

STR\$(cPx-R4),STR\$(cPy)~ %Point 2

!!b3

Add a quadratic bezier curve from the last point {(x1,y1)}, approaching control point (x2,y2), and ending at (x3,y3). If no \_MoveTo or \_LineTo call has been made for this contour, the first point is automatically set to (0,0) relativ to the placing coordinates.

Parameters:

x2 The x-coordinate of the control point on a quadratic curve

y2 The y-coordinate of the control point on a quadratic curve

x3 The x-coordinate of the end point on a quadratic curve

y3 The y-coordinate of the end point on a quadratic curve

!!e3

"\_QuadTo"~ %Quadratic Bezier Curve

STR\$(cPx-R4\*COS(Angle2\*PI()/180)),STR\$(cPy-R4\*SIN(Angle2\*PI()/180))~ %Point 2

STR\$(cPx),STR\$(cPy-R4)~ %Point 3

!!b4

Add a cubic bezier curve from the last point {(x1,y1)}, approaching control points (x2,y2) and (x3,y3), and ending at (x4,y4). If no \_MoveTo or \_LineTo call has been made for this contour, the first point is automatically set to (0,0) relativ to the placing coordinates.

Parameters:

x2 The x-coordinate of the 1st control point on a cubic curve

y2 The y-coordinate of the 1st control point on a cubic curve

x3 The x-coordinate of the 2nd control point on a cubic curve

y3 The y-coordinate of the 2nd control point on a cubic curve

x4 The x-coordinate of the end point on a cubic curve

y4 The y-coordinate of the end point on a cubic curve

!!e4

"\_CubicTo"~ %Cubic Bezier Curve

STR\$(cPx+R4\*COS(Angle3\*PI()/180)),STR\$(cPy-R4\*SIN(Angle3\*PI()/180))~ %Point 2

STR\$(cPx+R4\*COS(Angle1\*PI()/180)),STR\$(cPy-R4\*SIN(Angle1\*PI()/180))~ %Point 3

STR\$(cPx+R4),STR\$(cPy)~ %Point 4

!!b5

Append the specified arc to the path as a new contour. If the start of the path is different from the path's current last point, then an automatic \_LineTo is added to connect the current contour to the start of the arc. However, if the path is empty, then we call \_MoveTo with the first point of the arc.

Parameters:

Left, Top, Right, Bottom The bounds of oval defining shape and size of the arc.

StartAngle Starting angle (in degrees) where the arc begins

SweepAngle Sweep angle (in degrees) measured clockwise(!).

!!e5

"\_ArcTo"~

STR\$(cPx),STR\$(cPy-R4/4)~ %Left, Top

STR\$(cPx+R4),STR\$(cPy+R4/4)~ %Right, Bottom

STR\$(0),STR\$(180)~ %StartAngle, SweepAngle

!!b6

Append an arc to the path as a new contour. The arc is the specified by three coordinates. The start point is the last one of the path. The second and the third are the next coordinates.

A line from coordinates (1) to coordinates (3) is drawn if

coordinates (1), coordinates (2) and coordinates (3) are collinear, in this case also via (2);

coordinates (1) and coordinates (2) are equal;

coordinates (2) and coordinates (3) are equal or

coordinates (3) are missed, in this case to (2).

However, if the path is empty, then we call `_MoveTo` with the first point of the arc.

Parameters:

x2     The x-coordinate of the 1st control point on a cubic curve

y2     The y-coordinate of the 1st control point on a cubic curve

x3     The x-coordinate of the 2nd control point on a cubic curve

y3     The y-coordinate of the 2nd control point on a cubic curve

!!e6

"\_ArcSegTo"~

STR\$(cPx-R4/4),STR\$(cPy-R4/4)~ % Coordinates of the control point on an arc

STR\$(cPx-R4/2),STR\$(cPy)~ %Coordinates of the end point on an arc

!!b7

Close the current contour. If the current point is not equal to the first point of the contour, a line segment is automatically added.

!!e7

!"\_Close"~

!!b8

After `_End` the following items are ignored.

!!e8

"\_End"

!!b9

GR.PATH <obj\_nvar>, list\_pointer {x, y}

Parameters:

obj\_nvar     Object number

list\_pointer     List pointer of the list with curve type calls and coordinates

{x, y}     Placing coordinates

!!e9

GR.PATH gfirst, pathDraft1, 20, 0

GR.COLOR "\_Red"

GR.CIRCLE cir, cPx + 20, cPy, R4

GR.RENDER

DO

UNTIL 0

GitHub#56 1., 2., 3., 4.

**Gr.poly <obj\_nvar>, <list\_pointer\_nexp> {{{{ x, y}, <closed\_nexp>},  
<pointsPerPoly\_nexp>}, <paintPointerList\_nexp>}**

Creates an object that draws a closed polygon of any number of sides. The <obj\_nvar> returns the Object List object number for this polygon. This object will not be visible until the next **Gr.render**.

The list\_pointer is an expression that points to a List data structure. The list contains x, y coordinate pairs. The first coordinate pair defines the point at which the polygon drawing starts. Each subsequent coordinate pair defines a line drawn from the previous coordinate pair to this coordinate pair. A final line drawn from the last point back to the first closes the polygon. If you do not want to close the polygon <closed\_nexp> has to be 0. Default is <> 0. Note that the fill will still be drawn if a fill is specified in gr.color under Style (1 or 2).

If the optional x, y expression pair is present, the values will be added to each of the x and y coordinates of the list. This provides the ability to move the polygon array around the screen. The default x, y pair is 0,0. Negative values for x and y are valid.

The polygon line width, line color, alpha and fill are determined by previous **Gr.color** and **Gr.set.stroke** commands just like any other drawn object. These attributes are owned by the **poly** object, not by the list. If you use the same list in different **Gr.poly** commands, the color, stroke, etc., may be different.

You can change the polygon (add, delete, or move points) by directly manipulating the list with **List** commands. You can change to a different list of points using **Gr.Modify** with "list" as the tag parameter. Changes are not visible until the **Gr.render** command is called.

When you create a polygon with **Gr.poly** or attach a new list with **Gr.modify**, the list must have an even number of values and at least two coordinate pairs (four values). These rules are enforced with run-time errors. The rules cannot be enforced when you modify the list with **List** commands. Instead, if you have an odd number of coordinates, the last is ignored. If you have only one point, **Gr.render** draws nothing.

The **Gr.modify** parameters are "x", "y", "list", "paint" and "alpha".

See the Sample Program file, f30\_poly, for working examples of **Gr.poly**.

For fast access the list <list\_pointer\_nexp> can be divided by <pointsPerPoly\_nexp>. In this case more than one polygon can be created and the Gr.Paints of the polygons can be optional set by the list <paintPointerList\_nexp>. If this list has less entries than created polygons, the list will be start again at the first entry. Note, if you need only one color <paintPointerList\_nexp> is not needed. If you use **Gr.modify** later you can set also <paintPointerList\_nexp>. But there is only one way back if <paintPointerList\_nexp> has one or more entries with the same Gr.Paint.

Note, that changes in the List of <pointsPerPoly\_nexp> will be mapped by each following **Gr.render** command **without** a **Gr.modify** before.

### **Gr.arcpoly** <obj\_nvar>, <list\_pointer\_nexp> {{, x, y}, <closed\_nexp>}

Creates an object that draws a closed polygon of any number of sides as arc-segments or lines. The <obj\_nvar> returns the Object List object number for this polygon. This object will not be visible until the next **Gr.render**.

The list\_pointer is an expression that points to a List data structure. The list contains x, y coordinate pairs. The first coordinate pair defines the point at which the polygon drawing starts. Each subsequent pair of coordinates defines an arc or a line that is drawn from the previous pair of coordinates (1) via this pair of coordinates (2) to an end point defined by a further pair of coordinates (3).

A line from coordinates (1) to coordinates (3) is drawn if coordinates (1), coordinates (2) and coordinates (3) are collinear, in this case also via (2); coordinates (1) and coordinates (2) are equal; coordinates (2) and coordinates (3) are equal or coordinates (3) are missed, in this case to (2).

A final line drawn from the last point back to the first closes the polygon. If you do not want to close the polygon <closed\_nexp> has to be 0. Default is <> 0. Note that the fill will still be drawn if a fill is specified in gr.color under Style (1 or 2).

If the optional x, y expression pair is present, the values will be added to each of the x and y coordinates of the list. This provides the ability to move the polygon array around the screen. The default x, y pair is 0,0. Negative values for x and y are valid.

The polygon curve width, curve color, alpha and fill are determined by previous **Gr.color** and **Gr.set.stroke** commands just like any other drawn object. These attributes are owned by the **arcpoly** object, not by the list. If you use the same list in different **Gr.arcpoly** commands, the color, stroke, etc., may be different.

You can change the arc polygon (add, delete, or move points) by directly manipulating the list with **List** commands. You can change to a different list of points using **Gr.Modify** with "list" as the tag parameter. Changes are not visible until the **Gr.render** command is called. When you create an arc polygon with **Gr.arcpoly** or attach a new list with **Gr.modify**, the list must have an even number of values and at least two coordinate pairs (four values). These rules are enforced with run-time errors. The rules cannot be enforced when you modify the list with **List** commands. Instead, if you have an odd number of coordinates, the last is ignored. If you have only one point, **Gr.render** draws nothing. The **Gr.modify** parameters are "x", "y", "list", "paint" and "alpha".

See also WITHIN(), Gr.poly and Gr.path

### **Gr.rect** <obj\_nvar>, left, top, right, bottom{{, rx}, ry}

Creates a rectangle object. The rectangle will be located within the bounds of the parameters. The rectangle will or will not be filled depending upon the **Gr.color** style parameter. The <obj\_nvar> returns the **Display** List object number for this rectangle. This object will not be visible until the **Gr.render** command is called.

The **Gr.modify** parameters for **Gr.rect** are: "left", "top", "right", "bottom", "rx" and "ry".

GitHub#56 2.

### **Gr.get.bounds** <object\_ptr\_nexp>, <left\_nvar>, <top\_nvar>, <right\_nvar>, <bottom\_nvar>

Gets the boundary rectangle of a graphic object as it would be drawn on the screen. The returned coordinate values give you the dimensions of the bounding rectangle but not its location.

**Note objects turned by GR.ROTATE.START / GR.ROTATE.END do not return its new location and angle.**

If an object type is not supported all variables return 0.0.

Supported object types are:

arc, bitmap, circle, drawable, line, oval, point, rect and text

### **Gr.modify** <object\_ptr\_nexp> {, <tag\_sexp>, <value\_nexp | value\_sexp>}...

The value of the parameter named <tag\_sexp> in the Display List object <object\_ptr\_nvar> is changed to the value of the expression <value\_nexp> or <value\_sexp>. This command can change only one object at a time, but you may list as many tag/value pairs as you want.

With this command, you can change any of the parameters of any object in the Display List. The parameters you can change are given with the descriptions of the commands in this manual. In addition there are two general purpose parameters, "paint" and "alpha" (see below for details). You must provide parameter names that are valid for the specified object.

The results of **Gr.modify** commands will not be observed until a **Gr.render** command executes.



	TYPE	POSITION 1 (numeric)		POSITION 2 (numeric)		ANGLE/ RADIUS (numeric)	UNIQUE (various)	PAIN T (list ptr)	ALPHA (num)
SHAPES and OBJECTS	<a href="#">arc</a>	left	top	right	bottom	start_angle sweep_angle	fill_mode	paint	alpha
	<a href="#">arcpoly</a>	x	y				list	paint	alpha
	<a href="#">bitmap</a>	x	y				bitmap	paint	alpha
	<a href="#">circle</a>	x	y			radius		paint	alpha
	<a href="#">drawable</a>	left	top	right	bottom		drawable	paint	alpha
	<a href="#">line</a>	x1	y1	x2	y2			paint	alpha
	<a href="#">oval</a>	left	top	right	bottom			paint	alpha
	<a href="#">pixels</a>	x	y					paint	alpha
	<a href="#">point</a>	x	y					paint	alpha
	<a href="#">path</a>	x	y				list	paint	alpha
	<a href="#">poly</a>	x	y				list	paint	alpha
	<a href="#">rect</a>	left	top	right	bottom	rx, ry		paint	alpha
	<a href="#">text</a>	x	y				text	paint	alpha
MODIFIERS	<a href="#">clip</a>	left	top	right	bottom		RO	paint	alpha
	<a href="#">clipout</a>	left	top	right	bottom			paint	alpha
	<a href="#">group</a>						list	paint	alpha
	<a href="#">rotate</a>	x	y			angle		paint	alpha

#### TABLE NOTES:

- The **TYPE** column shows the string returned by **Gr.get.type** for each graphical object type.
- **Gr.get.position** returns the values in the **POSITION 1** columns.
- All table entries are **Gr.modify** tags (strings). Values of all the tags are numeric except for **"text"**.
- The values of tags in the **UNIQUE** column are either strings (**"text"**) or numbers with special interpretations. **"fill\_mode"** is a logical value. **"list"** is a pointer to a list of point coordinates. **"RO"** is a Region Operator as explained in **Gr.clip**.
- **"alpha"** is an integer value from 0 to 256, with 256 interpreted specially. See **General Purpose Parameters**, below.
- You can modify the **Gr.set.pixels** point-coordinates array directly. There is no **Gr.modify** tag.
- Modifiers with an underbar in front and uppercase letters like **"\_Paint"** are also accepted.

For example, suppose a bitmap object was created with **Gr.bitmap.draw BM\_ptr, galaxy\_ptr, 400, 120**.

Executing **gr.modify BM\_ptr, "x", 420** would move the bitmap from x = 400 to x = 420.

Executing **gr.modify BM\_ptr, "y", 200** would move the bitmap from y = 120 to y = 200.

Executing **gr.modify BM\_ptr, "x", 420, "y", 200** would change both x and y at the same time.

Executing **gr.modify BM\_ptr, "bitmap", Saturn\_ptr** would change the bitmap of an image of a (preloaded) Galaxy to the image of a (preloaded) Saturn.

**Gr.target.modify** <target\_sexp>, object\_ptr\_Array[], inp\_1\_Array[] {{, inp\_2\_Array[] }, inp\_3\_Array[], inp\_4\_Array[]}

Modifies graphic elements specified by the object\_ptr\_Array[]. The wished target is given by <target\_sexp>. See in this conjunction **Gr.modify** also.

Target	Element Types	inp_1_Array[]	inp_2_Array[]	inp_3_Array[]	inp_4_Array[]
_Position	Like Position 1 All(!) element types, but without <b>group</b>	X	X		
_Frame	Like Position 1 + 2 All element types with four parameters	X	X	X	X
_Move	Like Position 1 But moves the elements relative All(!) element types	X	X		
_Start_Sweep _Angles	Element type <b>arc</b>	X			
_Radius	Element type <b>circle</b>	X			
_Corner_Radii	Element type <b>rect</b>	X	X		
_Angle	Element type <b>rotate</b>	X			
_Fill_Mode	Element type <b>arc</b>	X			
_Bitmap	Element type <b>bitmap</b>	X			
_Drawable	Element type <b>drawable</b>	X			
_List	Element types <b>arcpoly, path, poly, group</b>	X			
_RO	Element type <b>RO</b>	X			
_Paint	Like Paint All element types	X			
_Alpha	Like Alpha All element types	X			

The results of **Gr.target.modify** commands will not be observed until a **Gr.render** command executes.

See also Gr.modify, Gr.move

Example:

```
Gr.target.modify "_Drawable", dAbles[], dAbleLeft[], dAbleTop[], dAbleRight[], dAbleBottom[]
```

**List.target.modify** <list\_ptr\_nexp>, <target\_sexp>, subObject\_ptr\_Array[], inp\_1\_Array[] {, inp\_2\_Array[], <points\_nexp>}

Modifies graphic elements specified by the subObject\_ptr\_Array[]. The wished target is given by <target\_sexp>. See in this conjunction **Gr.modify** also.

The optional inp\_2\_Array[] and <points\_nexp> are **only** used in conjunction with the Gr.poly enhancement to use more than one polygon with the same number of points by one command and one graphic object. In this case object\_ptr\_Array[] specifies the polygon object in this sub list.

If inp\_2\_Array[] and <points\_nexp> are specified the targets **\_Position** and **\_Move** can be changed.

If only inp\_1\_Array[] is specified the targets **\_Paint** and **\_Alpha** can be changed.

Command syntax of Gr.poly:

Gr.poly <obj\_nvar>, <list\_pointer\_nexp> {{{{, x, y}, <closed\_nexp>}, <pointsPerPoly\_nexp>}, <paintPointerList\_nexp>}

Target	Element Types	inp_1_Array[]	inp_2_Array[]
_Position	Like Position 1	X	X
_Move	Like Position 1 But moves the elements relative	X	X
_Paint	Like Paint	X	
_Alpha	Like Alpha	X	

The results of \_Position and \_Move have to be part of a **Gr.modify** command, but that will not be observed until a **Gr.render** command executes.

Note, the results of Paint and \_Alpha are mapped by each following **Gr.render** command **without** a **Gr.modify** before.

See also Gr.modify, Gr.move, Gr.paint

Example:

```
List.target.modify "_Move", listOfPoints, pointsOfPolygon, subObjectPtrArray[], xArray[], yArray[]
```

**GR\_COLLISION(<object\_1\_nvar>, <object\_2\_nvar>{,<dist\_1\_nvar>, <dist\_2\_nvar>})**

The variables <object\_1\_nvar> and <object\_2\_nvar> are the object pointers returned when the objects were created.

If the boundary boxes of the two objects overlap then the function will return true (not zero). If they do not overlap then the function will return false (zero).

Objects that may be tested for collision are: [arc](#), [bitmap](#), [circle](#), [drawable](#), [line](#), [oval](#), [point](#), [rect](#) and [text](#). In the case of a circle, an arc, [a line](#), an oval, or text, the object's rectangular boundary box is used for collision testing, not the actual drawn object.

**Note objects turned by GR.ROTATE.START / GR.ROTATE.END do not return its new location and angle. In this case the collision testing will fail.**

The arguments <dist\_1\_nvar> and <dist\_2\_nvar> allows to set back (-) or forth (+) the collision border. Take care that in a case of a backset your object is big enough.

Example:

A good example is a circle that touches a rectangle.

The circle has a diameter of 50 so a radius of 25.

With a circle backset of -25 (now a point) and a rectangle forthset of +25 the circle touches exactly the rectangle.

GR\_COLLISION(circle50, rectangle, -25, +25)

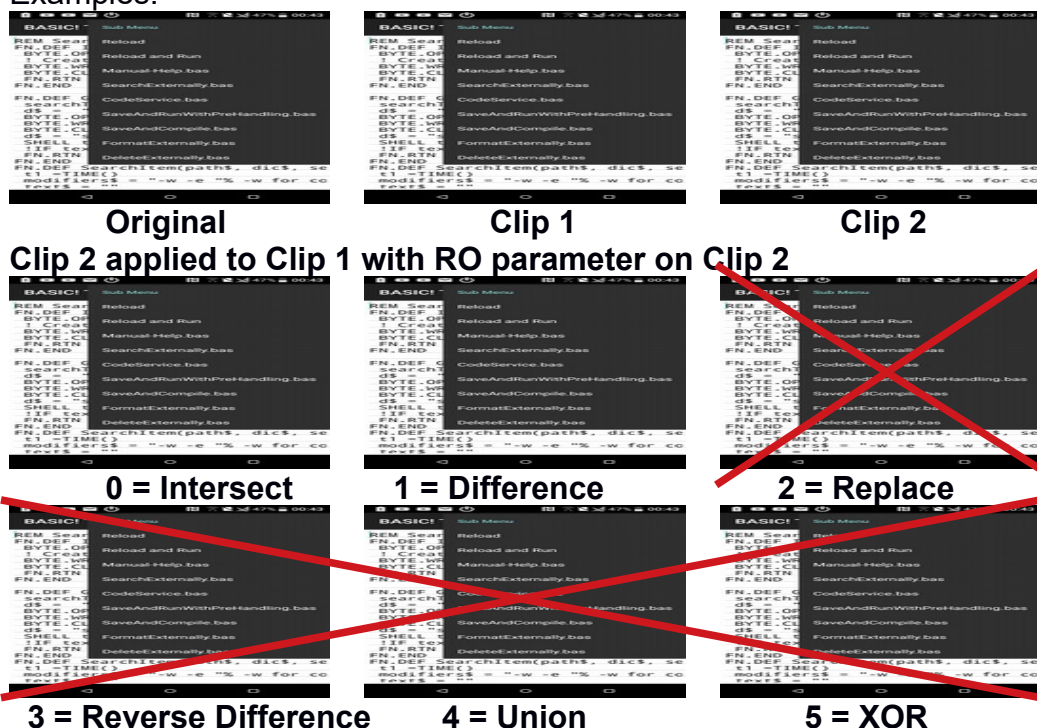
**Gr.clip** <object\_ptr\_nexp>, <left\_nexp>, <top\_nexp>, <right\_nexp>, <bottom\_nexp>{, <RO\_nexp>}

Objects that are drawn after this command is issued will be drawn only within the bounds (clipped) of the clip rectangle specified by the "left, top, right, bottom" numeric expressions. The final parameter is the Region Operator, <RO\_nexp>. The Region Operator prescribes how this clip will interact with everything else you are drawing on the screen or bitmap. If you issue more than one **Gr.clip** command, the RO prescribes the interaction between the current **Gr.clip** rectangle and the previous one. The RO values are:

<b>0</b>	<b>Intersect</b>
<b>1</b>	<b>Difference</b>
<b>2</b>	<b>Replace</b>
<b>3</b>	<b>Reverse Difference</b>
<b>4</b>	<b>Union</b>
<b>5</b>	<b>XOR</b>

The Region Operator parameter is optional. If it is omitted, the default action is **Intersect**.

Examples:



**Gr.clip** is a display list object. It can be modified with **Gr.modify**. The modify parameters are "left", "top", "right", "bottom", and "RO".

The **Gr.show** and **Gr.hide** commands can be used with the **Gr.clip** object.

RO values other than INTERSECT and DIFFERENCE have the ability to expand the clip, so values from 2 to 5 are ignored. The graphic mode clipping APIs are intended to only expand the clip as a result of a restore operation. This enables a view parent to clip a canvas to clearly define the maximal drawing area of its children. The recommended alternative calls are **Gr.clip** with <RO\_nexp> < 2 and **Gr.clipOut**.

See also **Gr.clipOut**, **Gr.color** with its optional xFermode

**Gr.clipOut** <object\_ptr\_nexp>, <left\_nexp>, <top\_nexp>, <right\_nexp>, <bottom\_nexp>

Sets the clip to the difference of the current clip and the specified rectangle, which is expressed by the "left, top, right, bottom" numeric expressions.

## Bitmap Commands

### Overview

When a bitmap is created, it is added to a list of bitmaps. Commands that create bitmaps return a pointer to the bitmap. The pointer is an index into the bitmap list. Your program works with the bitmap through the bitmap pointer.

If you want to draw the bitmap on the screen, you must add a graphical object to the Object List. The **Gr.bitmap.draw** command creates a graphical object that holds a pointer to the bitmap. Do not confuse the bitmap with the graphical object. You cannot use the Object Number to access the bitmap, and you cannot use the bitmap pointer to modify the graphical object.

Android devices limit the amount of memory available to your program. Bitmaps may use large blocks of memory, and so may exceed the application memory limit. If a command that creates a bitmap exceeds the limit, the bitmap is not created, and the command returns -1, an invalid bitmap pointer. Your program should test the bitmap pointer to find out if the bitmap was created. If the bitmap pointer is -1, you can call the **GETERROR\$()** function to get information about the error.

If a command exceeds the memory limit, but BASIC! does not catch the out-of-memory condition, your program terminates with an error message displayed on the Console screen. If you return the Editor, a line will be highlighted near the one that exceeded the memory limit. It may not be exactly the right line.

Bitmaps use four bytes of memory for each pixel. The amount of memory used depends only on the width and height of the bitmap. The bitmap is not compressed. When you load a bitmap from a file, the file is usually in a compressed format, so the bitmap will usually be larger than the file.

The counting of bitmap pointers was changed, because if it is possible, bitmaps will be overwritten if possible now. If you need automatically a new unique bitmap pointer number the value of the variable have to be 0 at command start. This is important when creating a bitmap for the first time. As a bitmap pointer you can use members of an array also. If a numeric variable or an element of an array has never been used before, the value is 0. If the value of the bitmap pointer variable is greater than 0 and has been mapped to the internal bitmap list, that bitmap will be overridden.

**Gr.bitmap.load** <bitmap\_ptr\_nvar>, <file\_name\_sexp>{{{{{, <wB\_nexp>}, <hB\_nexp>}, <cropX\_nexp>}, <cropY\_nexp>}, <cropW\_nexp>}, <cropH\_nexp>}, <bgColor\_sexp>}

Creates a bitmap from the file specified in the file\_name string expression. Returns a pointer to the created bitmap for use with other **Gr.bitmap** commands. If no bitmap is created, the returned bitmap pointer is -1. Call **GETERROR\$()** for information about the failure. Some of the possible causes are:

- The file or resource does not exist.
- There is not enough memory available to create the bitmap.

Bitmap image files are assumed to be located **as non source and non database files** in the "<pref base drive>/rfo-basic/data/" **main data** directory by default.

~~Note: You may include path fields in the file name. For example, "../Cougar.jpg" would cause BASIC! to look for Cougar.jpg in the top level directory of the base drive, usually the SD card. "images/Kitty.png" would cause BASIC! to look in the images(d) sub-directory of the "/sdcard/rfo-basic/data/" ("/sdcard/rfo-basic/data/images/Kitty.png").~~

Note: Bitmaps loaded with this command cannot be changed with the **Gr.bitmap.drawinto** command. To draw into an image loaded from a file, first create an empty bitmap then draw the loaded bitmap into the empty bitmap.

If a SVG (Scalable Vector Graphic) file is chosen the result based on the measurement in relation to 96 DPI. I.e. 5 cm /(2.54 cm/Inch) \* 96 (Dots/Inch) = 189 Dots

Rendering the SVG file the Android "\_Sans\_Serif" font is the default one. Special fonts like Arial, Verdana etc. are not supported.

The next arguments are optional.

The border arguments <wB\_nexp> and <hB\_nexp> specify the borders of the result.

The bitmap is scaled inside these borders, so that a square is a square and not a rectangle.

<wB_nexp>	<hB_nexp>	Result
0 (default)	0 (default)	In the original resolution
> 0	0	Scaled to the given width
0	> 0	Scaled to the given height

**Note that SVG files need to be converted large enough for the best quality.**

The arguments <cropX\_nexp>, <cropY\_nexp>, <cropW\_nexp>, <cropH\_nexp> are describing the position of the left (cropX) top (cropY) corner and the size (cropW, cropH) of the part to cut out. The defaults are 0,0,-1,-1. If cropW is -1 the right edge is limiting the cut out. If cropH is -1 the bottom edge is limiting the cut out. So the defaults return the full size. To prevent memory faults it is a good idea to crop at loading. Because older Android versions limit the maximum images size for loading to ≤ 2048 x ≤ 2048 pixel ≤ 12 MB RAM.

See also the <file\_name\_sexp> extension of Gr.bitmap.size about this behavior.

Mainly for buttons in conjunction with icons <bgColor\_sexp> sets the background color by the Gr.Paint color notation.

Example:

Gr.bitmap.size "blocks.svg", width, height

! Crop the right bottom quarter of "blocks.svg" in its original resolution + a new blue backgr.

Gr.bitmap.load bPtr, "blocks.svg",0,0, width/2, height/2, -1, -1, "\_10,Blue"

! Try also "\_Black" and "\_Red"



See also `Gr.bitmap.size`, `Gr.bitmap.crop`

### **Gr.bitmap.size <bitmap\_ptr\_nexp>|<file\_name\_sexp>, width, height**

Return the pixel width and height of the bitmap pointed to by <bitmap\_ptr\_nexp> or <file\_name\_sexp> into the width and height variables. SVG (Scalable Vector Graphic) files ending with \*.svg are also supported. In this case the measurement in relation to 96 DPI.

I.e.  $5 \text{ cm} / (2.54 \text{ cm/Inch}) * 96 \text{ (Dots/Inch)} = 189 \text{ Dots}$

The advantage of <file\_name\_sexp> is the detection of the size before loading. This prevents memory faults mostly on older devices.

The following should be noted about the memory requirements of bitmaps.

If you load bitmaps in their original size, the memory consumption can be very high, since also compressed Jpeg files are also decompressed when loading. The high resolutions of today's devices do the rest to make it a challenge.

Google therefore recommends only using or loading a reduced image or an image section. That's why `Gr.bitmap.size` got an extension to get the size by filename directly, so that the bitmap size can be determined before loading. To do this, however, the bitmap must be loaded internally. To avoid this with large bitmaps, you can make a preselection with `File.size`. For example, all compressed image files over 2 Mb will be shrunk by the following command.

`Gr.bitmap.load` has also an extension, by which an image file is reduced directly when loading or only a section of the image is taken.

See also `File.size`, `Gr.bitmap.load`

### **Gr.bitmap.clr <bitmap\_ptr\_nexp> {, <paint\_nexp>}**

Fills a bitmap completely with transparency without destroying it.

Anything already on the bitmap is cleared. There is no alpha blending.

Optional color from a paint.

This command is approximately ten times faster than the combination `Gr.bitmap.delete` with `Gr.bitmap.create`.

### **Gr.bitmap.save <bitmap\_ptr\_nvar>, <filename\_sexp>{, <quality\_nexp>}**

Saves the specified bitmap to a file. The default path is "<pref base drive>/rfo-basic/data/".

The file will be saved as a JPEG file if the filename ends in ".jpg".

The file will be saved as a PNG file if the filename ends in anything else (including ".png").

To <quality\_nexp>, the possible range is from 0 to 100. Default is 50.

### **Gr.bitmap.scale <new\_bitmap\_ptr\_nvar>, <bitmap\_ptr\_nexp>, width, height {, <smoothing\_lexp>}**

Scales a previously loaded bitmap (<bitmap\_ptr\_nexp>) to the specified width and height and creates a new bitmap <new\_bitmap\_ptr\_nvar>. The old bitmap still exists, it is not deleted. If there is not enough memory available to create the new bitmap, the returned bitmap pointer is -1. Call **GETERROR\$()** for information about the failure.

Negative values for width and height will cause the image to be flipped left to right or upside down.

Neither the width value nor the height value may be zero.

Use the optional smoothing logical expression (<smoothing\_lexp>) to request that the scaled image not be smoothed. If the expression is false (zero) then the image will not be smoothed. If the optional parameter is true (not zero) or not specified then the image will be smoothed.

**GR.bitmap.filter** <new\_bitmap\_ptr\_nvar>, <bitmap\_ptr\_nexp>, <bundl\_ptr\_nexp>  
Processes by filters a previously loaded bitmap (<bitmap\_ptr\_nexp>) specified by the <bundl\_ptr\_nexp> bundle and creates a new bitmap <new\_bitmap\_ptr\_nvar>. If the old bitmap still exists; it is not deleted.

Table of Bundle Keys		
Key	Value	Description
<b>_Filter</b>	_Binary _BlackFilter _Blur _Brightness _ColorRotation _ColorScale _EdgeDetection _Engrave _Flip _GammaCorrection _Hue _Invert _MaskPoly _ModifyOrientation _PolyToPoly _Rotate _RoundCorners _Saturation _Skew _Smooth _SnowEffect _Sharpen _UseColorMatrix _UseConvolutionMatrix _Watermark (String)	Set the filter type. <b>Always needed!</b>
	<b>_Binary</b>  The result is a Bitmap with only white and black pixels.	
		Default values.
<b>_BlackFilter</b>		
Base on randomizing image pixels, enhance the noise of darkness. The algorithm is to generate a threshold number (0-255), if all R,G,B values of a pixel are less than the threshold, then set the pixel to black.		
		Default values. It is a pixel by pixel function, so it needs more time.
<b>_Blur</b>		
Also called Gaussian Blur Effect		
<b>_Radius</b>	0.1 to 25 (numeric)	Default is 15.
<b>_Brightness</b> see <b>_ContrastBrightness</b>		

<b>_Value</b>	From -255 to +255 (numeric)	Default is 0. The result is limited to the possible minimum or maximum.
<b><u>_ColorRotation</u></b>		
Set the rotation on a color axis by the specified values.		
<b>_Axis</b>	_Red, _Green or _Blue (String)	Default is _Red.
<b>_Degrees</b>	0 to 360 (numeric)	Default is 0.
<b><u>_ColorScale</u></b>		
<b>_AlphaScale</b>	0 and positive numbers (numeric)	A value of 0 switches the color to off. 1 is identity and default. The result is limited to the possible maximum.
<b>_RedScale</b>	0 and positive numbers (numeric)	A value of 0 switches the color to off. 1 is identity and default. The result is limited to the possible maximum.
<b>_GreenScale</b>	0 and positive numbers (numeric)	A value of 0 switches the color to off. 1 is identity and default. The result is limited to the possible maximum.
<b>_BlueScale</b>	0 and positive numbers (numeric)	A value of 0 switches the color to off. 1 is identity and default. The result is limited to the possible maximum.
<b><u>_Contrast</u></b>		
<b>_Value</b>	0 and positive numbers (numeric)	A value of 0 sets the contrast to minimal. 1 is identity and default. The result is limited to the possible maximum.
<b><u>_ContrastBrightness</u></b>		
<p>Contrast is the difference in luminance and/or color that makes an object (or its representation in an image or display) distinguishable.</p> <p>In visual perception of the real world, contrast is determined by the difference in the color and brightness of the object and other objects within the same field of view. The concept of brightness is rather simple, increasing/decreasing value of each R, G, B channel together.</p> <p>+ By increasing: image results brighter.</p> <p>- By decreasing: image results darker.</p>		
<b>_BrightnessValue</b>	From -255 to +255 (numeric)	Default is 0. The result is limited to the possible minimum or maximum.
<b>_ContrastValue</b>	0 and positive numbers (numeric)	A value of 0 sets the contrast to minimal. 1 is identity and default. The result is limited to the possible maximum.
<b><u>_EdgeDetection</u></b>		
<b>_Level</b>	_Low, _Medium or _High (String)	Default is _Medium. The speed could be faster, but the results of the faster Android Render Script are strange.

<b>_Engrave</b>		
		Default values. It is a pixel by pixel function, so it needs more time.
<b>_Hue</b>		
Translates the colors within the color wheel by the specified angle. White, gray and black are not affected.		
<b>_Degrees</b>	0 to 360 (numeric)	Default is 0.
<b>_Flip</b>		
<b>_Horizontal</b>	0 or 1 (numeric)	Default is 0.
<b>_Vertical</b>	0 or 1 (numeric)	Default is 0.
<b>_GammaCorrection</b>		
<b>_Gamma</b>	0 to 8 (numeric)	1 is identity and default. Correction by a 1 / gamma value. Values below 1 provide darker results. Values greater than 1 provide brighter results. It is a pixel by pixel function, so it needs more time.
<b>_Invert</b>		
Inverts the bitmap from a positive into a negative or a negative into a positive one.		
		Default values.
<b>_MaskPoly</b>		
Masks the bitmap with a polygon.		
<b>_SourcePoints</b>	Array (numeric)	Default is {0,0,0,0,0,0,0,0} Array of polygon nodes positions as x,y pairs. You can use polygons with three or more nodes.
<b>_Type</b>	_In or _Out (String)	Default is _In. If the _Type is _In the area inside the polygon is returned as <b>transparent</b> . The argument _Out stands for the area outside.
<b>_ModifyOrientation</b>		
Modifies the orientation of the current bitmap specified by the EXIF of a bitmap given by an URL.		
<b>_ImageUrl</b>	(String)	
<b>_PolyToPoly</b>		
Crops from a source bitmap a part defined by a polygon with <u>three</u> or <u>four</u> nodes. The result is described by a destination polygon with the same number of nodes. The order of the nodes are beginning at the left top corner in a clockwise turned direction. Sometimes we get <b>in trouble</b> , if we use a polygon with <b>four</b> nodes and a small source bitmap. In this case try to scale the source bitmap to double size or more.		

<b>_SourcePoints</b>	Array (numeric)	Default is {0,0,0,0,0,0,0,0} Array of polygon nodes positions as x,y pairs. You can use polygons with three or four nodes.
<b>_DestinationPoints</b>	Array (numeric)	Default is {0,0,0,0,0,0,0,0} Array of polygon nodes positions as x,y pairs. You can use polygons with three or four nodes.
<b>_Mask</b>	0 or 1 (numeric)	Default = 1 Masks the source bitmap by the _SourcePoints to get a transparent area outside the polygon.
<b>_Debug</b>	0 or 1 (numeric)	Debug results will be printed at the console.
<b>_Rotate</b>		
<b>_Degrees</b>	0 to 360 (numeric)	Default is 0. Keep in mind, that an angle $\neq 0$ , 180 perhaps also 90 and 270 (width equals high) will make the bitmap larger. Areas of the resulting four triangles are transparent.
<b>_RoundCorners</b>		
<b>_Radius</b>	0 and positive numbers (numeric)	Default is 0.
<b>_Saturation</b>		
<b>_Value</b>	0 and positive numbers (numeric)	A value of 0 maps the color to <b>gray-scale</b> . 1 is identity and default. Values > 1 <b>boost</b> the colors. The result is limited to the possible maximum.
<b>_Sharpen</b>		
<b>_Level</b>	_Low, _Medium or _High (String)	Default is _Medium.
<b>_Skew</b>		
<b>_DeltaX</b>	0 and positive numbers (numeric)	Default is 0. Sets the x size of the skew.
<b>_DeltaY</b>	0 and positive numbers (numeric)	Default is 0. Sets the y size of the skew.
<b>_AtX</b>	0 and positive numbers (numeric)	Default is 0. Sets the x position of the skew.
<b>_AtY</b>	0 and positive numbers (numeric)	Default is 0. Sets the y position of the skew.
<b>_Smooth</b>		
<b>_Value</b>		Default is 0.
<b>_SnowEffect</b>		
Opposite of Black Filter Sets all pixels having R,G,B values to the max of 255 when they are greater than threshold.		

		Default values. It is a pixel by pixel function, so it needs more time.
<b>_UseColorMatrix</b>		
<b>_ColorMatrix</b>	Example: Array.Load colorMatrix[],~ %[ -1, 0, 0, 0, 255,~ 0, -1, 0, 0, 255,~ 0, 0, -1, 0, 255,~ 0, 0, 0, 1, 0 % ] (numeric array)	
<b>_UseConvolutionMatrix</b>		
<b>_ConvolutionMatrix</b>	Example for _3x3 Matrix: Array.Load convolutionMatrix[],~ %[ -0.15, -0.15, -0.15,~ -0.15, 2.2, -0.15,~ -0.15, -0.15, -0.15 % ] (numeric array)	
<b>_Type</b>	_3x3 or _5x5 (String)	Default is _3x3.
<b>_ColorSpace</b>	_ARGB_8888 or _RGB_565 (String)	Default is _ARGB_8888. Use the _RGB_565 option with care, because there is no chance to handle an error!
<b>_Watermark</b>		
Gives your image a watermark.		
<b>_Text</b>	(String)	Default is "". Text of the watermark.
<b>_Color</b>	{Alpha,}Red,Green,Blue (comma delimited string) or _{Alpha,}ColorName ({comma delim.} string) or #{hn}hnhnhn (hex. string)	Default is "".
<b>_AtX</b>	0 and positive numbers (numeric)	Default is 0. Sets the x position of the watermark.
<b>_AtY</b>	0 and positive numbers (numeric)	Default is 0. Sets the y position of the watermark.
<b>_Size</b>	0 and positive numbers (numeric)	Default is 10. Font size of the watermark.
<b>_Underline</b>	0 or 1 (numeric)	Default is 0. If > 0 the watermark has a underline.

Example 1:

```
GR.OPEN "_White", 1, 1
GR.BITMAP.LOAD bPtr1, "cartman.png"
```

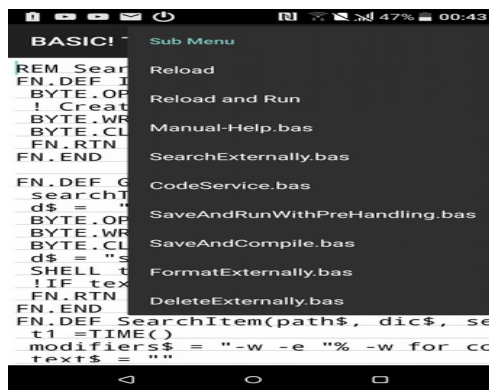
```
BUNDLE.PUT bndPtr1, "_Filter", "_Hue"  
counter = 0  
DO  
  BUNDLE.PUT bndPtr1, "_Degrees", counter  
  GR.BITMAP.SCALE bPtr2, bPtr1, 600,600  
  GR.BITMAP.FILTER bPtr3, bPtr2, bndPtr1  
  GR.BITMAP.DRAW oPtr1, bPtr3, 250, 250  
  GR.RENDER  
  counter = counter +5  
UNTIL counter = 365
```



**Gr.bitmap.crop** <new\_bitmap\_ptr\_nvar>, <source\_bitmap\_ptr\_nexp>, <x\_nexp>, <y\_nexp>, <width\_nexp>, <height\_nexp>

Creates a cropped copy of an existing source bitmap specified by <source\_bitmap\_ptr\_nexp>. The source bitmap is unaffected; a rectangular section is copied into a new bitmap. A pointer to the new bitmap is returned in <new\_bitmap\_ptr\_nvar>. If there is not enough memory available to create the new bitmap, the returned bitmap pointer is -1. Call **GETERROR\$()** for information about the failure.

The <x\_nexp>, <y\_nexp> pair specifies the point within the source bitmap that the crop is to start at. The <width\_nexp>, <height\_nexp> pair defines the size of the rectangular region to crop.



To check the bounds, use as an example:

```
GR.BITMAP.SIZE bp, wx, hy % 48, 48
x = 10
y = 20
width = 10
height = 15
IF x < 0 | (x + width) > wx | width < 1 | y < 0 | (y + height) > hy | height < 1
    PRINT "Error: The crop frame is out of bounds"
END
END IF
GR.BITMAP.CROP newBp, bp, x, y
```

**Gr.bitmap.drawinto.end** {<legacy\_mode\_nexp>}

End the draw-into-bitmap mode. Subsequent draw commands will place the objects into the display list for rendering on the screen. If you wish to display the drawn-into bitmap on the screen, issue a **Bitmap.draw** command for that bitmap.

For legacy reasons <legacy\_mode\_nexp> suppresses a runtime error, if Gr.bitmap.drawinto.start is not executed before.

### **Gr.bitmap.get.histogram <bitmap\_ptr\_nexp>, alpha[], red[], green[], blue[]**

The bitmap pointer is specified by <bitmap\_ptr\_nexp>. The command returns the color channel specific histogram arrays alpha[], red[], green[] and blue[]. Each array contains the sum of pixels with the same intensity. If 47 pixel have the red intensity of 233, red[234] returns 47. 234 because BASIC! arrays are starting with an index of 1 instead of 0.

### **Gr.bitmap.get.selected.pixarr <bitmap\_ptr\_nexp>, x[], y[], alpha[], red[], green[], blue[][, colorNumbers[]]**

The bitmap pointer is specified by <bitmap\_ptr\_nexp>. The command returns the color channel specific pixel arrays alpha[], red[], green[] and blue[] at the pixel points defined by the x[] and y[] arrays. The optional colorNumbers[] returns an array of system color numbers.

If a pixel point is outside of the bitmap the color channels return -1. In case of colorNumbers[] Infinity will be returned.

(IF colorNumbers[1] < VAL("Infinity") THEN PRINT "Within".)

Keep in mind, that the first left top pixel point is 0 (x[]), 0 (y[]). The last right bottom pixel point is width -1, height -1.

### **Gr.bitmap.get.pixarr <bitmap\_ptr\_nexp>, alpha[], red[], green[], blue[][, colorNumbers[]]**

The bitmap pointer is specified by <bitmap\_ptr\_nexp>. The command returns the color channel specific pixel arrays alpha[], red[], green[] and blue[]. The optional colorNumbers[] returns an array of system color numbers. The order is column by column.

Keep in mind, that the first left top pixel point is 1, 1 instead of 0, 0, because the BASIC! Array index base is 1. The last right bottom pixel point is width, height.

Example of a bitmap with 5 pixels width and 2 pixels height:

y\x	1	2	3	4	5
1	0	200	40	60	80
2	100	30	50	70	90

[3,2] = 50

Example:

```
GR.OPEN
GR.BITMAP.LOAD bPtr, "cartman.png"
GR.BITMAP.GET.PIXARR bPtr, alpha[], red[], green[], blue[]
ARRAY.DIMS alpha[], dims[]
DEBUG.ON
DEBUG.DUMP.ARRAY dims[]
```

See also `GR.bitmap.filter` with the key `_UseColorMatrix`

### **Gr.bitmap.set.pixarr <bitmap\_ptr\_nvar>, alpha[], red[], green[], blue[]**

Creates a bitmap by the bitmap pointer <bitmap\_ptr\_nvar> and sets the pixels by the color channel specific pixel arrays. The order is column by column specified by alpha[]. An alpha[] array dimension-ed by [5,2] returns a bitmap with 5 pixels width and 2 pixels height. The defaults of the arrays red[], green[], blue[] are filled with 255. Values < 0 are changed to 0 and values > 255 to 255.

Example of a bitmap with 5 pixels width and 2 pixels height:

$y \backslash x$	1	2	3	4	5
1	0	200	40	60	80
2	100	30	50	70	90

[3,2] = 50

Example:

```
! Bitmap 100*40 with random colored pixels
b = 100 : h = 40 : bS = b * h
DIM bSArray[bS]
ARRAY.FILL bSArray[], 255
ARRAY.LOAD d[], b, h % 100*40 = 4000
ARRAY.TO.DIMS bSArray[], d[], alpha[] % Only alpha[] dims need to be specified
ARRAY.RND red[], bS, 0, 255
ARRAY.RND green[], bS, 0, 255
ARRAY.RND blue[], bS, 0, 255
GR.BITMAP.SET.PIXARR nRndPtr, alpha[], red[], green[], blue[]
GR.BITMAP.DRAW rndPtr, nRndPtr, 200, 600
GR.RENDER
```

See also GR.bitmap.filter with the key \_UseColorMatrix

## Drawable Commands

### Overview

Beginning with Android **9.0** / Pie API **28** animated drawables are supported.

Unfortunately was the file access to all types of animated drawables not be back-ported for earlier versions until now.

Drawable is a parent type for drawing and animating graphics.

Child types are bitmap- or vector graphics are animated or fixed.

Supported image file types are BMP, PNG, JPEG, WEBP, GIF or HEIF.

If the encoded image is an animated GIF or WEBP, the animation has to be started by `GR.drawable.start`.

Until including Android **8.1** / Oreo API **28** the image file types BMP, PNG, JPEG and GIF are supported.

But it is not possible to convert a simple bitmap to an animated GIF.

When a drawable is created, it is added to a list of drawables. Commands that create drawables return a pointer to the drawable. The pointer is an index into the drawable list.

Your program works with the drawable through the drawable pointer.

If you want to draw the drawable on the screen, you must add a graphical object to the Object List. The **Gr.drawable.draw** command creates a graphical object that holds a pointer to the drawable. Do not confuse the drawable with the graphical object. You cannot use the Object Number to access the drawable, and you cannot use the drawable pointer to modify the graphical object.

Android devices limit the amount of memory available to your program. drawables may use large blocks of memory, and so may exceed the application memory limit. If a command that creates a drawable exceeds the limit, the drawable is not created, and the command returns -1, an invalid drawable pointer. Your program should test the drawable pointer to find out if the drawable was created. If the drawable pointer is -1, you can call the **GETERROR\$()** function to get information about the error.

If a command exceeds the memory limit, but if BASIC! does not catch the out-of-memory condition, your program terminates with an error message displayed on the Console screen. If you return the Editor, a line will be highlighted near the one that exceeded the memory limit. It may not be exactly the right line.

Drawables can use up to four bytes of memory for each pixel. The amount of memory used depends mainly on the width and height of the drawable. The drawable is not compressed. When you load a drawable from a file, the file is usually in a compressed format, so the drawable will usually be larger than the file.

The counting of drawable pointers was changed, because if it is possible, drawables will be overwritten if possible now. If you need automatically a new unique drawable pointer number the value of the variable have to be 0 at command start. This is important when creating a drawable for the first time. As a drawable pointer you can use members of an array also. If a numeric variable or an element of an array has never been used before, the value is 0.

If the value of the drawable pointer variable is greater than 0 and has been mapped to the internal drawable list, that drawable will be overridden.

### **Gr.drawable.load <drawable\_ptr\_nvar>, <file\_name\_sexp>**

Creates a drawable from the file specified in the file\_name string expression. Returns a pointer to the created drawable for use with other **Gr.drawable** commands. If no drawable is created, the returned drawable pointer is -1. Call **GETERROR\$()** for information about the failure. Some of the possible causes are:

- The file or resource does not exist.
- There is not enough memory available to create the drawable.

drawable image files are assumed to be located in the "<pref base drive>/rfo-basic/data/" directory.

Note: Drawables loaded with this command cannot be changed directly. To draw into an image loaded from a file, first create an empty bitmap then draw the loaded drawable into the empty bitmap by Gr.bitmap.drawinto.start.

See also Gr.bitmap.load

### **Gr.drawable.fromBitmap <drawable\_ptr\_nvar>, <bitmap\_ptr\_nexp>**

Creates a drawable from a given bitmap specified by the bitmap pointer <bitmap\_ptr\_nexp>. Returns a pointer to the created drawable for use with other **Gr.drawable** commands.

Note: The bitmap referenced by <bitmap\_ptr\_nexp> must not be deleted.

See also Gr.bitmap.load

### **Gr.drawable.draw <object\_ptr\_nvar>, <drawable\_ptr\_nexp>, left, top, right, bottom**

Creates a graphical object that contains a drawable and inserts the object into the Object List. The drawable is specified by the drawable pointer <drawable\_ptr\_nexp>. The drawable will be drawn within the bounds of the parameters.. The command returns the Object List object number of the graphical object in the <object\_ptr\_nvar> variable. This object will not be visible until the **Gr.render** command is called.

The alpha value of the latest **Gr.color** will determine the transparency of the drawable.

The **Gr.modify** parameters for **Gr.drawable.draw** are: "drawable", "left", "top", "right" and "bottom".

The use of borders instead of coordinates is due to the ability to resize the images more easily.

See also Gr.bitmap.draw

### **Gr.drawable.start <drawable\_ptr\_nvar>**

Starts the animation of the drawable <drawable\_ptr\_nvar> if possible. A following Gr.render is needed to start.

### **Gr.drawable.stop <drawable\_ptr\_nvar>**

Stops the animation of the drawable <drawable\_ptr\_nvar> if possible. A following Gr.render is needed to stop.

### **Gr.drawable.delete** <drawable\_ptr\_nexp>

Deletes an existing drawable. The drawable's memory is returned to the system. This does not destroy any graphical object that points to the drawable. If you do not **Gr.hide** such objects, or remove them from the Display List, you will get a run-time error from the next **Gr.render** command.

See also `Gr.bitmap.delete`

### **Gr.cls** {<clear\_bitmaps/drawables\_nexp>}

Clears the graphics screen. Deletes all previously drawn objects; all existing object references are invalid. Deletes all existing Paints and resets all **Gr.color** or **Gr.text** {size|align|bold|strike|underline|skew} settings. Disposes of the current Object List and Display List and creates a new Initial Display List.

Note: bitmaps and drawables are not deleted. They will not be drawn because no graphical objects point to them, but the bitmaps or drawables still exist. Variables that point to them remain valid.

If the optional <clear\_bitmaps/drawables\_nexp> is > 0, then all bitmaps and drawables are deleted also.

In this case all variables that point to them are not valid.

The **Gr.render** command must be called to make the cleared screen visible to the user.

### **Gr.statusbar** <height\_nvar> {, showing\_lvar}

Returns information about the Status Bar. If the **height** variable <height\_nvar> is present, it is set to the nominal height of the Status Bar. If the **showing** flag <showing\_lvar> is present, it is set to **0** (false, not showing) or **1** (true, showing) based on how Graphics Mode was opened.

The parameters are both optional. If you omit the first parameter but use the second, you must keep the comma.

**Gr.screen** <width\_nvar>, <height\_nvar>{[{{{, density\\_nvar](#) }, isRound\_lvar> }, <layout[ ]>, <insets[ ]>, <bounds[ ]>}

Returns the screen's width and height, and optionally its density, in the numeric variables. The density, in dots per inch (dpi), is a standardized Android density value (usually 120, 160, 240 or [480 dpi](#)), and not necessarily the real physical density of the screen.

If a **Gr.orientation** command changes the orientation, the width and height values from a previous **Gr.screen** command are invalid.

Android's orientation-change animation takes time. You may need to wait for a second or so after **Gr.open** or **Gr.orientation** before executing **Gr.screen**, otherwise the width and height values may be set before the orientation change is complete.

[Placing this command behind OnGrScreen: should solve this issue.](#)

**Gr.screen** returns a subset of the information returned by the newer **Screen** command.

If the display is round isRound returns [1](#).

The <layout[ ]> array returns the bounds of the current display view. [Left, Top, Right, Bottom].

**It is strongly recommended, to use the <layout[ ]> array for graphic-screen size-calculation!**

If the device has one or two notches and Android ≥ 9:

The <insets[ ]> array returns the insets of the current display view. [Left, Top, Right, Bottom].

The <bounds[ ]> array returns the bounds of the notch(es). [Left, Top, Right, Bottom{,Left, Top, Right, Bottom} ].

Example:

If a nutch with the dimensions 150 x 100 is in the upper left corner

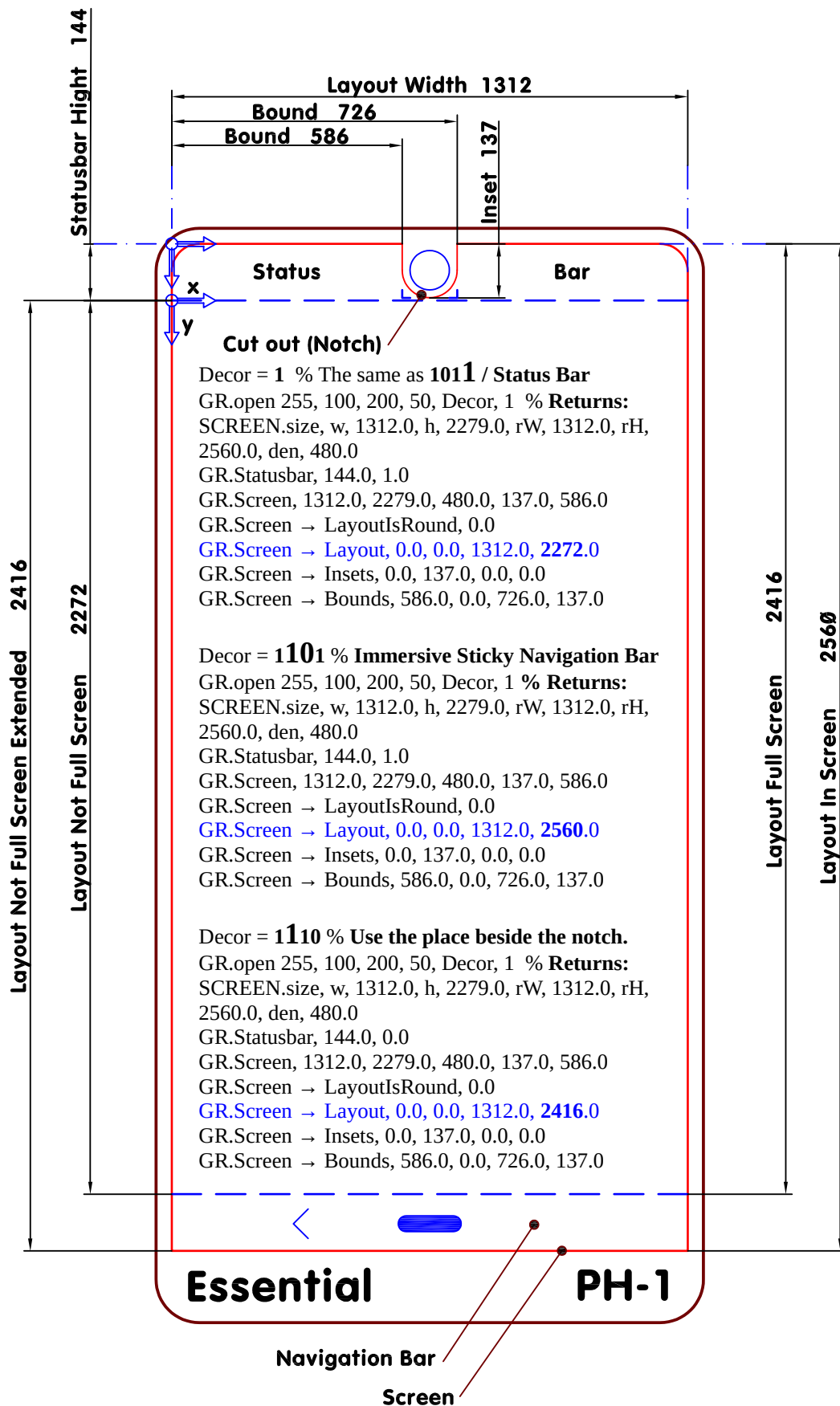
insets[] returns [150, 100, 0, 0]

and

bounds[] returns [0, 0, 150, 100].

See also [OnGrScreen](#).; [Screen](#)

Table of Decor Options				
Decor = [ 1 ] [ ] [ ] [ ]		0	1	2
[ 1 ]	Place holder		In any case	
[ ]	Use the place beside the cut out	No	Yes	
[ ]	Navigation bar	Immersive Sticky Mode (Android 7+, Translucent is the fall back.)	Yes	Translucent
[ ]	Status bar	No	Yes	
Decor = [ ]	Status bar	No	Yes	





### OnGrScreen:

Interrupt label that traps on receive a screen layout change. BASIC! executes the statements following the **OnGrScreen:** label until it reaches a **GR.ongerscreen.resume**.

Example:

```
FN.DEF SetViewSizes (globals)
  GR.SCREEN w, h, density, isRound , layout[ ]
  w = (layout[3] - layout[1])
  h = (layout[4] - layout[2])
  layoutRatio = w/h
  BUNDLE.PUT globals, "layoutW", w
  BUNDLE.PUT globals, "layoutH", h
  FN.RTN layoutRatio
FN.END

FN.DEF DrawLandscape (globals)
  BUNDLE.GET globals, "layoutW", w
  BUNDLE.GET globals, "layoutH", h
  PRINT "Draw Landscape"
  FN.RTN 1
FN.END

FN.DEF DrawPortrait (globals)
  BUNDLE.GET globals, "layoutW", w
  BUNDLE.GET globals, "layoutH", h
  PRINT "Draw Portrait"
  FN.RTN 1
FN.END

BUNDLE.CREATE globals
PRINT globals

PRINT "A OliBasic Solution"
GR.OPEN "_Blue", , -1 % Orientation changeable

DO
  PAUSE 20
UNTIL 0

ONGRSCREEN:  % Interrupt will be also send at GR.OPEN
  layoutRatio = SetViewSizes (globals)
  IF layoutRatio > 1
    DrawLandscape (globals)
  ELSE
    DrawPortrait (globals)
  ENDIF
GR.ONGRSCREEN.RESUME
```

## GR.ongrscreen.resume

Resumes execution at the point in the BASIC! program where the **OnGrScreen:** interrupt occurred.

## Gr.scale x\_factor, y\_factor{{, x\_distance{, y\_distance}}

Scale all drawing commands by the numeric x and y scale factors and translates optional by the numeric x and y distance. This command is provided to allow you to draw in a device-independent manner and then scale and translate the drawing to the actual size of the screen that your program is running on.

The translation will be executed before scaling. If you need translation after scaling multiply factor with distance.

For example:

```
! Set the device independent sizes
di_height = 480
di_width = 800

! Get the actual width and height
gr.open          % defaults: white, no status bar, landscape
gr.screen actual_w, actual_h

! Calculate the scale factors
scale_width = actual_w / di_width
scale_height = actual_h / di_height

! Set the scale
gr.scale scale_width, scale_height
```

Now, start drawing based upon di\_height and di\_width. The drawings will be scaled to fit the device running the program.

## Gr.scale.touch x\_factor, y\_factor{{{, x\_distance}, y\_distance}, <reverse\_nexp>}

Scale all touch commands by the numeric x and y scale factors and translates optional by the numeric x and y distance. This command is provided to allow you to touch in a device-independent manner, scaling and translating the touches to the actual size of the screen that your program is running on. If <reverse\_nexp> set to 1 you can insert the same values used in Gr.scale to compensate its scale and translation values. Default is 0, in this case you have to compute the values yourself.

See also Gr.scale, Gr.touch, Gr.last.touch

## Gr.array.touch Array[], <count\_nvar>

Returns all available touched points as an array of pairs (x, y), independent if these are touched or moved. The variable <count\_nvar> returns the number of points. This command should be placed behind onTimer: and before Timer.resume, for best stability. Set.Timer is recommended with 10 or more.

See also Gr.scale, Gr.touch, Gr.last.touch, Gr.scale.touch, Gr.list.touch

### **Gr.list.touch <listX(Y)\_pointer\_nexp>, {<listY\_pointer\_nexp>}, <count\_nvar>**

Returns all available touched points, independent if these are touched or moved and as one list of pairs (x, y) if <listY\_pointer\_nexp> is not defined. When <listY\_pointer\_nexp> is used it's list contains the Y value of the point and the list <listX(Y)\_pointer\_nexp> the X value. The variable <count\_nvar> returns the number of points. This command should be placed behind onTimer: and before Timer resume, for best stability. Set.Timer is recommended with 10 or more.

See also      Gr.scale, Gr.touch, Gr.last.touch, Gr.scale.touch, Gr.array.touch,  
Timer.set, Sched.set (If you need a second timer.)

### **Gr.last.touch <last\_index\_nvar>, <x\_nvar>, <y\_nvar>**

Returns the last touched index with <last\_index\_nvar> with the (x, y) coordinates of the touch. If the screen is not currently touched, Touched returns false (0) with the (x,y) coordinates of the last previous touch. If the screen has never been touched, the x and y variables are left unchanged.

The returned values are relative to the actual screen size. Thus if you scaled the screen, you need to scale the returned parameters in the opposite direction.

See also Gr.scale, Gr.touch, Gr.scale.touch

### Gr.touch touched, x, y

Tests for a touch on the graphics screen. If the screen is being touched, Touched is returned as true (not 0) with the (x,y) coordinates of the touch. If the screen is not currently touched, Touched returns false (0) with the (x,y) coordinates of the last previous touch. If the screen has never been touched, the x and y variables are left unchanged. The command continues to return true as long as the screen remains touched.

If you want to detect a single short tap, after detecting the touch, you should loop until touched is false.

```
DO
GR.TOUCH touched, x, y
UNTIL touched

! Touch detected, now wait for
! finger lifted
DO
GR.TOUCH touched, x, y
UNTIL !touched
```

The returned values are relative to the actual screen size. If you have scaled the screen then you need to similarly scale the returned parameters. If the parameters that you used in **Gr.scale** were scale\_x and scale\_y (**Gr.scale scale\_x, scale\_y**) then divide the returned x and y by those same values.

```
GR.TOUCH touched, x, y
Xscaled = x / scale_x
Yscaled = y / scale_y
```

### Gr.touch2 touched, x, y

The same as **Gr.touch** except that it reports on second simultaneous touch of the screen.

#### WARNING

Use this command with caution, as the event handler works like a tennis player batting against a much too fast set ball-machine. If possible, use Gr.array.touch or Gr.list.touch, as these commands recognize the x-y-status of one or more fingers at the **same** time.

Use instead if possible Gr.array.touch or Gr.list.touch!

### Gr.bounded.touch2 touched, left, top, right, bottom

The same as **Gr.bounded.touch** except that it reports on second simultaneous touch of the screen.

See also the warning at Gr.touch2

Use instead of Gr.bounded.touch and Gr.bounded.touch2 if possible Gr.list.touch and Within()!

### OnGrTouch:

Interrupt label that traps any touch **down** on the Graphics screen (see "Interrupt Labels"). BASIC! executes the statements following the **OnGrTouch:** label until it reaches a **Gr.onGrTouch.resume** command.

To detect touches on the Output Console (not in Graphics mode), use **OnConsoleTouch:**.

If you want to detect a double tap use:

```
grTapCounter = 0
PRINT "GR Double Tap Example"
DO
UNTIL 0
END "Bye...!"

OnTimer:
IF grTapCounter = 1
PRINT "Only One Tap!"
TIMER.CLEAR
grTapCounter = 0
ENDIF
TIMER.RESUME
OnGrTouch:
grTapCounter++
IF grTapCounter = 1
TIMER.CLEAR
TIMER.SET 500
ENDIF
IF grTapCounter = 2
PRINT "Double Tap!"
TIMER.CLEAR %Needed, if you will not end execution.
grTapCounter = 0 %Needed, if you will not end execution.
PAUSE 2000
END "Bye...!"
ENDIF
Gr.onGrTouch.resume
```

Note: OliBasic's SCHED.SET command can work as a timer, too.

### OnGrTouchMove:

Interrupt label that traps any touch **move** on the Graphics screen (see "Interrupt Labels"). BASIC! executes the statements following the **OnGrTouchMove:** label until it reaches a **Gr.onGrTouchMove.resume** command.

### Gr.onGrTouchMove.resume

Resumes execution at the point in the BASIC! program where the **OnGrTouchMove:** interrupt occurred.

### OnGrTouchUp:

Interrupt label that traps any touch **up** on the Graphics screen (see "Interrupt Labels"). BASIC! executes the statements following the **OnGrTouchUp:** label until it reaches a **Gr.onGrTouchUp.resume** command.

## Gr.onGrTouchUp.resume

Resumes execution at the point in the BASIC! program where the **OnGrTouchUp:** interrupt occurred.

Example:

```
FN.DEF swipeDirection(bx, by, ex, ey, t1, t2, vMinQ)
  sQ = (ey-by)^2 + (ex-bx)^2
  t = t2 - t1
  vQ = sQ/t
  mDirection = ATAN2(ey-by,ex-bx)
  mDirection = ROUND(mDirection/PI()*2)
  IF mDirection = -1 THEN mDirection = 3
  IF vQ > vMinQ
    FN.RTN ABS(mDirection)
  ELSE
    FN.RTN -1
  ENDIF
FN.END
! 200 from 255 alpha; 1000 no status bar and no navigation bar
GR.OPEN "_200,DarkBlue",1000,-1
GR.SCREEN osx, osy, density, isR, lo[]
! It is strongly recommended, to use the <layout[ ]> array
sx = lo[3]-lo[1] : sy = lo[4]-lo[2] %[Left, Top, Right, Bottom].
! Device independent; v^2, because we save the square root later
vMinQ = (4 * density / 160)^2
GR.COLOR "_Red",1
GR.CIRCLE goc,sx/2,sy/2,sx/50
GR.SET.STROKE 5
GR.TEXT.SIZE sy/30
mDir$ = "Direction = "
GR.TEXT.DRAW vd,20,sy/18, mDir$

GR.RENDER
mPause = 150
DO
  PAUSE mPause
UNTIL 0

ONGRTOUCH:
  mPause = 20
  mSwipe = 1 : t1 = clock()
  GR.TOUCH touched, mSwipeXb, mSwipeYb
  ! PRINT "TOUCH"
  GR.ONGRTOUCH.RESUME

ONGRTOUCHMOVE:
  mSwipe ++
  GR.TOUCH touched, cx, cy
  GR.MODIFY goc,"x",cx,"y",cy
  GR.RENDER
  ! PRINT "MOVE"
  GR.ONGRTOUCHMOVE.RESUME

ONGRTOUCHUP:
  IF mSwipe > 2
```

```

GR.TOUCH touched, mSwipeXe, mSwipeYe
mDirecetion = swipeDirection(mSwipeXb, mSwipeYb, mSwipeXe, mSwipeYe, t1, ~
clock(), vMinQ)
! PRINT "SWIPE: "; mDirecetion
GR.MODIFY vd,"text", mDir$ + INT$(mDirecetion)
GR.RENDER
ENDIF
mPause = 100
! PRINT "UP"
GR.ONGRTOUCHUP.RESUME

```

## Graphics Setup Commands

**Gr.open {{alpha}{, red}{, green}{, blue}{, <Decors\_nexp>}{, <Orientation\_nexp>}}  
{,<Camera\_nexp>}**

Opens the Graphics Screen and puts BASIC! into Graphics Mode. The color values become the background color of the graphics screen. The default color is opaque white (255,255,255,255).

All parameters are optional; use commas to indicate omitted parameters (see Optional Parameters).

Each of the four color components is a numeric expression with a value from 0 through 255. If a value is outside of this range, only the last eight bits of the value are used; for example, 257 and 1025 are the same as 1. If any color parameter is omitted, it is set to 255. **You are able to use color definitions like "\_127,Green", "\_Blue", "#ff008080" etc. also. See the color definition pages at the end of this appendix.**

Beginning with Android 7 and 9 some things have changed around screen decors like the status bar.

Starting with Android 7 the background of the status bar is the same as the graphic screen. If you switch the status bar to on and the background is white you see no text or icon. So you should darken the background in this case in the bounds of the status bar. See also Gr.Screen.

Starting with Android 9 the operating system supports display cutouts also called notches. The Status Bar Text will be shown on the graphics screen if the <Decors\_nexp> is 1 or 1XX[1]. If the <Decors\_nexp> is 0, 1XX[0] or the <Decors\_nexp> is not present, the Status Bar Text will not be shown.

Ending with Android 6 the background of the status bar is darken to Black by default. But keep in mind, that the coordinate system starts still in left, top display corner. Thus you have to move your graphic objects down if needed.

Beginning with Android 9 you can deal with the bounds of the cut outs.

If <Decors\_nexp> 1X[0]X the navigation bar will be in the Immersive Sticky Mode, thus the navigation bar is only visible, if you stroke a touch from the outside into the screen.

If <Decors\_nexp> 1X[1]X the navigation bar will be displayed in the normal mode.

If <Decors\_nexp> 1X[2]X the navigation bar will be in the Translucent Mode, thus the navigation bar and the content behind are visible.

If <Decors\_nexp> 1[0]XX the drawing layout will be shrinked by the insets of the cut out(s). In this case the coordinate system starts in left, top **shrinked layout** corner.

If <Decors\_nexp> is 1[1]XX the drawing layout is like the display layout. If a cut out or round display edges are within your graphic, it will not be displayed (because no display area) but this area will still be saved in a screenshot. See also Gr.Screen.

If <Decors\_nexp> is **negative**, Gr.open expects a layout bundle defined by items of the second following table like Gr.open "\_Blue", - myLayoutBundle, -1.

### For graphic-screen size-calculation and layout dimensions refer to the command **Gr.screen!**

The orientation upon opening graphics will be determined by the <Orientation\_nexp> value. <Orientation\_nexp> values are the same as values for the **Gr.orientation** command (see below). If the <Orientation\_nexp> is not present, the default orientation is Landscape. **It is strongly recommended to insert the start orientation in the Gr.open command, because to prevent trouble in conjunction Gr.orientation and Gr.screen directly behind Gr.open.**

The <Camera\_nexp> parameter sets an optional camera view behind the graphics screen.

- 0 No camera view in background (default)
- 1 Camera view with camera on back
- 2 Camera view with camera in front
- If > as number of existing cameras, the camera with the highest id is used.

Keep in mind, that Android's numbering starts with 0 BASIC! starts with 1.

Today only Android 6+ is supported.

If no CAMERA permission is granted, <Camera\_nexp> falls back to 0.

**Note, if <Camera\_nexp> is > 0 and some commands like select, dialog.select, dialog message and text.input are used at running in graphics mode, the *program will be halted*. So use this commands outside the graphic mode if a camera is turned to on. Keep in mind, that Orientation -1 is not allowed if Camera > 0.**

If the app goes in background in this GR.Open camera mode, you have to use GR.close after detecting with OnBackground: and Background().

See also Gr.screen, Gr.statusbar

Example:

```
FN.DEF OpenGraphicDisplay()
  sBr = 1101
  SCREEN.SIZE size[], realsize[], density
  ori = 0
  IF size[1] < size[2] THEN ori = 1
  cam = 1
  GR.OPEN 0, 10, 0, 0, sBr, ori , cam

  GR.CAMERA.GETPARAM p$
  p$ = REPLACE$(p$, ";;","\n")
  PRINT p$
  s$ = "effect=mono"
  GR.CAMERA.SETPARAM s$,1

  fl = 3
  GR.CAMERA.FLASH fl

  GR.STATUSBAR sHeight, sE
  GR.SCREEN w, h
  GR.COLOR 255,250,0,0,2
  yFromTop = sHeight + 50
  GR.RECT rc1, 50, (sHeight*sE)+ 50, w-50, (sHeight*sE) + 150, 10
  GR.RENDER
```



FN.RTN 1

FN.END

OpenGraphicDisplay()

mGr = 1

bg = 0

zoomFactor = 100

GR.CAMERA.ZOOM zoomFactor

zoomFactorMax = zoomFactor

zoomDirection = 1

DO

PAUSE 100

UNTIL 0

ONGRTOUCH:

IF zoomFactor > (zoomFactorMax-0.1) THEN zoomDirection = -1

zoomFactor = zoomFactor + 0.2 \* zoomDirection

IF zoomFactor < 1 THEN zoomDirection = 1 : zoomFactor = 1

GR.CAMERA.ZOOM zoomFactor

BIGD.ROUND zoomFactor\$, STR\$(zoomFactor), 1, "HU"

sel = -400

DIALOG.MESSAGE "Zoom Factor", zoomFactor\$, sel

GR.ONGRTOUCH.RESUME

ONBACKGROUND:

IF BACKGROUND()

IF mGr = 1

GR.CLOSE

mGr=0

ENDIF

bg = 1

ELSE

IF bg = 1

OpenGraphicDisplay()

mGr = 1

bg = 0

ENDIF

ENDIF

BACKGROUND.RESUME


Table of Decor Options

Decor = [ 1 ] [ ] [ ] [ ]		0	1	2
[ 1 ]	Place holder		In any case	
[ ]	Use the place beside the cut out	No	Yes	
[ ]	Navigation bar	Immersive Sticky Mode (Android 7+. Translucent is the fall back.)	Yes	Translucent
[ ]	Status bar	No	Yes	

Decor = [ ]	Status bar	No	Yes	

The optional options bundle (-) <Decors\_nexp> controls the layouts of the Action and Navigation bars. That means, a negative <Decors\_nexp> is interpreted as a bundle pointer.

Table of Layout Control Options		
Key	Value	Description
<b>_ShowActionbar</b>	0 or 1 (numeric)	If 1 Show the Action bar if it is not currently showing. It is needed to show titles and to change the background color of the Statusbar. If 0 (default) Hide the Actionbar if it is currently activated.
<b>_Title</b>	String	Set the action bar's title.
<b>_Subtitle</b>	String	Set the action bar's subtitle.
<b>_TitleShow</b>	0 or 1 (numeric)	If 1 Show the Action bar if it is not currently showing. It will resize application content to fit the new space available. If 0 (default) Hide the Actionbar if it is currently showing. It will resize application content to fit the new space available.
<b>_TitleIcon</b>	Icon file path	Add a large icon to the notification content view. <a href="http://romannurik.github.io/AndroidAssetStudio/index.html">http://romannurik.github.io/AndroidAssetStudio/index.html</a>
<b>_TitleHomeEnabled</b>	0 or 1 (numeric)	Set whether to include the application home accordance in the action bar. Home is presented as an activity icon. Have to be 1 if you want to show the icon. Have to be 0 if you want to hide the icon. The default setting is API dependent.
<b>_TitleBackground</b>	Background file path	

<b>_TitleHtml</b>	0 or 1 (numeric)	<p>Returns displayable styled text from the provided HTML string. But not all tags are supported.</p>  <p>Uses parts of TagSoup library to handle real HTML, including all of the brokenness found in the wild.</p> <p> <b>&lt;b&gt;</b>  <b>&lt;big&gt;</b>  <b>&lt;font size="..." color="..." face="..."&gt;</b>  <b>&lt;h1&gt;</b>, <b>&lt;h2&gt;</b>, <b>&lt;h3&gt;</b>, <b>&lt;h4&gt;</b>, <b>&lt;h5&gt;</b>,  <b>&lt;h6&gt;</b>  <b>&lt;i&gt;</b>  <b>&lt;small&gt;</b>  <b>&lt;strike&gt;? &lt; A.7</b>  <b>&lt;strong&gt;</b>  <b>&lt;sub&gt;</b>  <b>&lt;sup&gt;</b>  <b>&lt;tt&gt;?</b>  <b>&lt;u&gt;</b> </p> <p><b>Replace Space with &amp;#160, &amp; with &amp;amp, &lt; with &amp;lt, &gt; with &amp;gt, " with &amp;quot if necessary.</b></p> <p>Usable for Title and Subtitle. Keep in mind that the Action bar height will not be expanded.</p>
<b>_ShowStatusbar</b>	0, 1 or 2 (numeric)	<p>If 1 (default) The Status bar will be displayed.</p> <p>If 2 The Status bar will be transparent displayed. Min. Lollipop 5.0 (API 21)</p> <p>If 0 The Status bar will be hidden to the background. Min. Nougat 7.0 (API 24) Will be switched to option 2 or 1 if the current API level is lower.</p>
<b>_StatusbarColor</b>	{Alpha,}Red,Green,Blue (comma delimited string) or _{Alpha,}ColorName ({{comma delim.} string) or #{hn}hnhnhn (hex. string)	<p>Min. Lollipop 5.0 (API 21)</p> <p>Note, the ActionBar has to be activated by <b>_ShowActionBar</b>.</p>

<b>_StatusBarLight</b>	0 or 1 (numeric)	<p>If 0 (default) The Status bar background is dark. In this case the <b>bar content</b> will be <b>light</b>.</p> <p>If 1 The Status bar background is light. In this case the <b>bar content</b> will be <b>dark</b>.</p> <p>Min. Lollipop 5.0 (API 21)</p>
<b>_ShowNavigationbar</b>	0, 1 or 2 (numeric)	<p>If 1 (default) The Navigation bar will be displayed.</p> <p>If 2 The Navigation bar will be transparent displayed.</p> <p>Min. Lollipop 5.0 (API 21)</p> <p>If 0 The Navigation bar will be hidden to the background.</p> <p>Min. Nougat 7.0 (API 24)</p> <p>Will be switched to option 2 or 1 if the current API level is lower.</p>
<b>_NavigationbarColor</b>	{Alpha,}Red,Green,Blue (comma delimited string) or _{Alpha,}ColorName ({{comma delim.} string) or #{hn}hnhnhn (hex. string)	Min. Lollipop 5.0 (API 21)
<b>_NavigationbarLight</b>	0 or 1 (numeric)	<p>If 0 (default) The Navigation bar background is dark. In this case the <b>bar content</b> will be <b>light</b>.</p> <p>If 1 The Navigation bar background is light. In this case the <b>bar content</b> will be <b>dark</b>.</p> <p>Min. Lollipop 5.0 (API 21)</p>
<b>_Menu</b>	Menu Bundle Pointer	<p>Creates menu entries. A successful selection will be returned as a human readable JSON string.</p> <p>See the example at Console.title for more details.</p>
<b>_ExtendBesideNotch</b>	0 or 1 (numeric)	<p>If 0 (default) The space beside the notch is not used for graphics.</p> <p>If 1 The space beside the notch is used also.</p> <p>Min. Pie 9.0 (API 28)</p>

<b>_CameraViewBounds</b>	left,top,right,bottom (comma delimited string)	Defines the bounds of a background camera view in pixels.
--------------------------	---	--

## IS\_GR ()

IS\_GR returns the graphic mode status. If it is 1 the graphic mode is enabled. Is 0 returned the graphic mode is not open.

See also Console.Save

### **GR.set.acceleration <mode\_nvar>**

This command overwrites the preference settings. Use this command with care.

For starters is a good place directly behind the Gr.open command.

If <mode\_nvar> = 0 (SOFTWARE): The graphic view is rendered in software into a bitmap.

If <mode\_nvar> = 1 (HARDWARE): The view is rendered in hardware into a hardware texture if the application is hardware accelerated.

If <mode\_nvar> = 2 (NONE): The view is rendered normally and is not backed by an off-screen buffer.

The default behavior is the mode gotten from the APK-xml or set in the preferences.

The advantage of hardware acceleration is not only the speed, the power consumption is much lower too.

If you cannot use the hardware acceleration, because some details are not displayed, proceed as follows. Create your graphic, if nothing happened in one or two seconds, take a screenshot and display the saved bitmap on top if necessary. Switch to hardware acceleration and wait for an event by OnGrTouch:. Now switch back to software rendering.

For more information consult also:

<https://developer.android.com/guide/topics/graphics/hardware-accel#java>

Gr.render equals the Java function invalidate().

Look also under Unsupported Drawing Operations.

For Android 6+

See also Timer.Set, onTimer:, Sched.Set, onSched:, Gr.ScreenToBitmap, Gr.hide

### Gr.set.cap {{<cap\_nexp>},{<paint\_nexp>}}

Sets the line caps of objects drawn after this command is issued.

The opportunities of <cap\_nexp> are shown in this table:

Value	Meaning	DescriptionThe stroke ends with the path, and does not project beyond it.
0	BUTT	The stroke ends with the path, and does not project beyond it.
1	ROUND	The stroke projects out as a semicircle, with the center at the end of the path.
2	SQUARE	The stroke projects out as a square, with the center at the end of the path.

Example:

```
GR.OPEN 80,0,0,0,0,1
GR.ARC GR.SET.CAP 2
GR.SET.STROKE 80
GR.COLOR 255,255,0,0
GR.LINE nn,100,100,400,400
GR.COLOR 255,255,0,0,0 % Red with fill mode = 0, now the default one
GR.ARC cc,100,500,400,800, 0, -270, 0
```

```
GR.SET.STROKE 60
GR.SET.CAP 1
GR.COLOR "_DarkBlue"
GR.LINE nn,100,100,400,400
GR.ARC cc,100,500,400,800, 0, -270, 0
```

```
GR.SET.STROKE 10
GR.COLOR "_Green"
GR.SET.CAP 0
GR.LINE nn,100,100,400,400
GR.ARC cc,100,500,400,800, 0, -270, 0
```

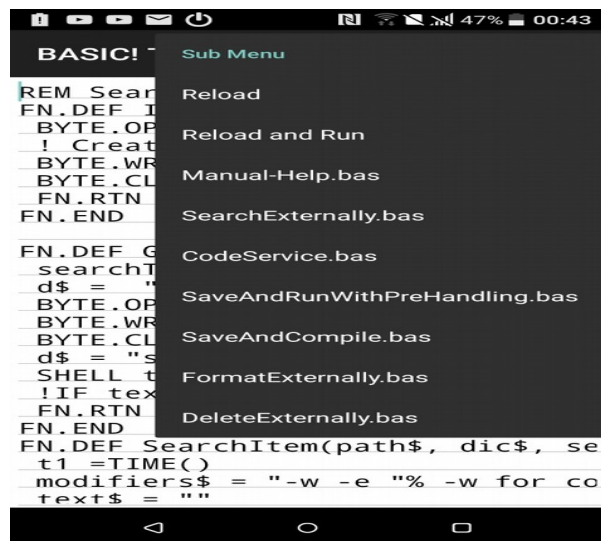
```
GR.SET.STROKE 80
GR.LINE nn,500,100,700,300
GR.LINE nn,700,300,900,100
```

```
GR.SET.CAP 1
GR.COLOR "_DarkBlue"
GR.LINE nn,500,300,700,500
GR.LINE nn,700,500,900,300
```

```
GR.SET.STROKE 10
GR.SET.CAP 0
GR.COLOR "_Red"
GR.LINE nn,500,100,500,300
GR.LINE nn,900,100,900,300
```

```
GR.RENDER
```

```
DO:UNTIL 0
```



### **Gr.color** {{alpha}{, red}{, green}{, blue}{, style}{, paint}{, xFermode}}

Sets the color and style for drawing objects. There are two ways to use this command.

- *Basic usage*: ignore the optional <paint> parameter. The new color and style will be used for whatever graphical objects are subsequently drawn until the next **Gr.color** command is executed.
- *Advanced usage*: The "basic usage" of this command always creates a new Paint. If you prefer, you can use the <paint> parameter to specify an existing Paint. The **Gr.color** command sets the color and style of that Paint, changing the appearance of any graphical object to which it is attached. The current Paint is not changed. See "Paints Advanced Usage" above and the example below.

All of the parameters are optional. If a color component or the style is omitted, that component is left unchanged. For example, **Gr.color** ,,0 sets only green to 0, leaving alpha, red, blue, and style as they were. Use commas to indicate omitted parameters (see Optional Parameters).

Each of the four color components (alpha, red, green, blue) is a numeric expression with a value from 0 through 255. If a value is outside of this range, only the last eight bits of the value are used; for example, 257 and 1025 are both the same as 1.

**You are able to use color definitions like "\_127,Green", "\_Blue", "#ff008080" etc. also. See the color definition pages at the end of this appendix.**

**Also possible "\_{<alpha>},HSV<hue[0...360]>{,<saturation [0...1]>},<valueOfBrightness [0...1]>". The string "\_200,HSV240" returns the color blue by a color wheel angle of 240 degrees with alpha of 200.**

See also

[https://en.wikipedia.org/wiki/HSL\\_and\\_HSV](https://en.wikipedia.org/wiki/HSL_and_HSV), [https://en.wikipedia.org/wiki/Color\\_wheel](https://en.wikipedia.org/wiki/Color_wheel).

The style parameter, is a numeric expression that determines the stroking and filling of objects. The effect of this parameter is explained in detail in the "Style" sections, see above. The possible values for <style\_nexp> are shown in this table:

Value	Meaning	Description
0	STROKE	Geometry and text drawn with this style will be stroked (outlined), respecting the stroke-related fields on the paint.
1	FILL	Geometry and text drawn with this style will be filled, ignoring all stroke-related settings in the paint.
2	STROKE_AND_FILL	Geometry and text drawn with this style will be filled and stroked at the same time, respecting the stroke-related fields on the paint.

If you specify a value other than -1, 0, 1, or 2, then the style is set to 2. If you specify a style of -1, the style is left unchanged, just as if the style parameter were omitted. If you never set a style, the **default** value is **1, FILL**.

You can change the stroke weight with commands such as **Gr.set.stroke** (see below) and the various text style commands.

Example:

```
GR.OPEN
! basic usage
GR.COLOR ,0,0,255,2          % opaque blue, stroke and fill
GR.RECT r1, 50,50,100,100    % draw two squares
GR.RECT r2, 100,100,150,150
GR.COLOR 128,255,0,0          % half-transparent red
GR.RECT r3, 75,75,125,125     % draw an overlapping square
```



GR.RENDER : PAUSE 2000

! advanced usage

GR.GET.VALUE r1, "paint", p

GR.COLOR 255,0,255,0,,p

GR.RENDER : PAUSE 2000

GR.RECT r4, 125,125,175,175

GR.RENDER : PAUSE 2000

GR.CLOSE : END

% get index of first Paint

% change that Paint's color to opaque green

% both r1 and r2 change

% use current Paint, unchanged

% still draws half-transparent red

The argument xFermode sets the Porter-Duff Compositing and Blend Modes.  
If you want to use it directly, the Background on Gr.open the alpha has to be set to 0.  
For a mask use Gr.bitmap.drawinto.start with Gr.bitmap.drawinto.end.

Index	Mode	Formula
-1	NORMAL	<b>Default</b>
0	CLEAR	[Sa, Sc] Destination pixels covered by the source are cleared to 0.
1	SRC	[Da, Dc] The source pixels replace the destination pixels.
2	DST	[Sa + (1 - Sa)*Da, Rc = Sc + (1 - Sa)*Dc] The source pixels are discarded, leaving the destination intact.
3	SRC_OVER	[Sa + (1 - Sa)*Da, Rc = Dc + (1 - Da)*Sc] The source pixels are drawn over the destination pixels.
4	DST_OVER	[Sa * Da, Sc * Da] The source pixels are drawn behind the destination pixels.
5	SRC_IN	[Sa * Da, Sa * Dc] Keeps the source pixels that cover the destination pixels, discards the remaining source and destination pixels.
6	DST_IN	[Sa * (1 - Da), Sc * (1 - Da)] Keeps the destination pixels that cover source pixels, discards the remaining source and destination pixels.
7	SRC_OUT	[Da * (1 - Sa), Dc * (1 - Sa)] Keeps the source pixels that do not cover destination pixels. Discards source pixels that cover destination pixels. Discards all destination pixels.
8	DST_OUT	[Da, Sc * Da + (1 - Sa) * Dc] Keeps the destination pixels that are not covered by source pixels. Discards destination pixels that are covered by source pixels. Discards all source pixels.
9	SRC_ATOP	[Sa, Sa * Dc + Sc * (1 - Da)] Discards the source pixels that do not cover destination pixels. Draws remaining source pixels over destination pixels.
10	DST_ATOP	[Sa + Da - 2 * Sa * Da, Sc * (1 - Da) + (1 - Sa) * Dc] Discards the destination pixels that are not covered by source pixels. Draws remaining destination pixels over

		source pixels.
11	XOR	$[Sa + Da - Sa * Da, Sc * (1 - Da) + Dc * (1 - Sa) + \min(Sc, Dc)]$ Discards the source and destination pixels where source pixels cover destination pixels. Draws remaining source pixels.
12	DARKEN	$[Sa + Da - Sa * Da, Sc * (1 - Da) + Dc * (1 - Sa) + \max(Sc, Dc)]$ Retains the smallest component of the source and destination pixels.
13	LIGHTEN	$[Sa * Da, Sc * Dc]$ Retains the largest component of the source and destination pixel.
14	MULTIPLY	$[Sa + Da - Sa * Da, Sc + Dc - Sc * Dc]$ Multiplies the source and destination pixels.
15	SCREEN	Saturate(S + D) Adds the source and destination pixels, then subtracts the source pixels multiplied by the destination.
16	ADD	Adds the source pixels to the destination pixels and saturates the result.
17	OVERLAY	Multiplies or screens the source and destination depending on the destination color.

For more information see:

<https://developer.android.com/reference/android/graphics/PorterDuff.Mode.html>

Example:

```
GR.OPEN 0
GR.COLOR 150, 0, 155,0,1, , -1
GR.RECT destinationR, 20,80,400,600
GR.COLOR 255, 255, 0,0,1, , 0
GR.CIRCLE sourceC, 300, 300, 200
GR.RENDER
DO
UNTIL 0
```

## COLOR(<sexp>)

Returns the system color number of a color string expression.

{Alpha,}Red,Green,Blue (comma delimited string) or _{Alpha,}ColorName ({comma delim.} string) or #{hn}hnhnhn (hex. string)	Sets the color. Default is "255,0,0,0" or "_Black", "_HSV240" or "#ff000000" See also Gr.color
---	---

## COLOR\$(<nexp>)

Returns a color string expression of a system color number of the type  
"0-255, 0-255, 0-255, 0-255" (alpha, red, green, blue)

### Gr.orientation <nexp> depriciate

The value of the <nexp> sets the orientation of screen as follows:

- 1 = Orientation depends upon the sensors.
- 0 = Orientation is forced to Landscape.
- 1 = Orientation is forced to Portrait.
- 2 = Orientation is forced to Reverse Landscape.
- 3 = Orientation is forced to Reverse Portrait.

You can monitor changes in orientation by reading the screen width and height using the **Gr.screen** or **Screen** commands.












Keep in mind, that the camera orientation in the GR.Open camera mode is not changed.
















## Paint Commands





















### Gr.paint.set <bundle\_nexp>{, <paint\_nexp>}

Creates a new or overwrites an existing Paint with arguments provided by the given bundle <bundle\_nexp>.

The optional <paint\_nexp> selects the Paint definition to overwrite. If <paint\_nexp> is -1 a new Paint definition is created. The default is 0 which use the current Paint.

Table of Bundle Keys			
	Key	Value	Description
	<b>_TextAlign</b>	_Left, _Center or _Right	Align the text relative to the (x,y) coordinates given in the Gr.text.draw command.
	<b>_TextBold</b>	_On or _Off	
	<b>_TextSkew</b>	(numeric)	Set the paint's horizontal skew factor for text. The default value is 0. For approximating oblique text, use values around -0.25.
	<b>_TextSize</b>	(numeric)	Sets the text size.
	<b>_TextScaleX</b>	(numeric)	Set the paint's horizontal scale factor for text. The default value is 1.0. Values > 1.0 will stretch the text wider. Values < 1.0 will stretch the text narrower.
	<b>_TextLetterSpace</b>	(numeric)	Set the paint's letter-spacing for text. The default value is 0. The value is in 'EM' units. Typical values for slight expansion will be around 0.05. Negative values tighten text. Only Android 5+
	<b>_TextWordSpace</b>	(numeric)	Set the paint's extra word-spacing for text. Increases the white space width between words with the given amount of pixels. The default value is 0. Only Android 5+
	Font, style and typeface are also handled by Paint pointers. But please use Gr.text.setfont and Gr.text.typeface <b>behind Gr.paint.set.</b>		
	<b>_Color</b>	{Alpha,}Red,Green,Blue (comma delimited string) or _{Alpha,}ColorName ({comma delim.} string) or #{hn}hnhnhn (hex. string)	Sets the color. Default is "255,0,0,0" or "_Black", "_HSV240" or "#ff000000" See also Gr.color
	<b>_Antialias</b>	_On or _Off	Default is "_On". See also Gr.set.antialias
	<b>_Style</b>	_Fill, _Stroke or _Fill&Stroke	Default is "_Fill". See also Gr.color

	<b>_StrokeWidth</b>	(numeric)	Sets the stroke width. Default is 0.
	<b>_StrokeCap</b>	_CapButt, _CapRound or _CapSquare	Sets the line caps. Default is "_CapButt". See also Gr.set.cap
	<b>_DashPathEffect</b>	_On or _Off	See also Gr.set.dashpatheffect
	<b>_PathPattern</b>	list pointer (numeric)	The intervals list must contain an even number of entries ( $\geq 2$ ), with the even indices specifying the "on" intervals, and the odd indices specifying the "off" intervals.
	<b>_Phase</b>	(numeric)	Phase is an offset into the intervals list (modifies the sum of all of the intervals).
	<b>_Xfermode</b>	_Normal, _Clear, _Src, _Dst, _Src_Over, _Dst_Over, _Src_In, _Dst_In, _Src_Out, _Dst_Out, _Src_Atop, _Dst_Atop, _Xor, _Darken, _Lighten, _Multiply, _Screen, _Add or _Overlay	Sets the Porter-Duff Compositing and Blend Modes. Default is "_Normal". See also Gr.color
	<b>_Shader</b>	_LinearGradient, _Pattern, _RadialGradient or _SweepGradient (String)	Creates a Shader
	<b>_LinearGradient</b>		
	<b>_X0</b>	(numeric)	Default is 0. Upper left corner in X direction
	<b>_Y0</b>	(numeric)	Default is 0. Upper left corner in Y direction
	<b>_X1</b>	(numeric)	Default is 0. Lower right corner in X direction
	<b>_Y1</b>	(numeric)	Default is 0. Lower right corner in Y direction
	<b>_ColorArray</b>	Array of {Alpha,}Red,Green,Blue (comma delimited string) or _{Alpha,}ColorName ({comma delim.} string) or #{hn}hnhnhn (hex. string)	Default is {"_Black", "_White"}  An array of colors that creates the gradient from the top to the bottom.
	<b>_PositionArray</b>	Array (numeric)	Default is none. Array of positions
	<b>_TileMode</b>	_Clamp, _Mirror or _Repeat (String)	Default is "_Mirror". Sets the tile mode

	<b>_Degree</b>	(numeric)	Default is 0. Turns the Shader clockwise.
	<b>_Pattern</b> (Bitmap Shader)		
	<b>_Bitmap</b>	Bitmap object number (numeric)	Source bitmap
	<b>_Degree</b>	(numeric)	Default is 0. Turns the Shader clockwise.
	<b>_TileModeX</b>	_Clamp, _Mirror or _Repeat (String)	Default is "_Repeat". Sets the tile mode in X direction
	<b>_TileModeY</b>	_Clamp, _Mirror or _Repeat (String)	Default is "_Repeat". Sets the tile mode in Y direction
	<b>_RadialGradient</b>		
	<b>_CenterX</b>	(numeric)	Default is 0. Center point in X direction
	<b>_CenterY</b>	(numeric)	Default is 0. Center point in Y direction
	<b>_Radius</b>	> 0 (numeric)	
	<b>_ColorArray</b>	Array of {Alpha,}Red,Green,Blue (comma delimited string) or _{Alpha,}ColorName ({comma delim.} string) or #{hn}hnhnhn (hex. string)	Default is {"_Black", "_White"}
	<b>_StopArray</b>	Array (numeric)	Default is none. Array of stop points
	<b>_TileMode</b>	_Clamp, _Mirror or _Repeat (String)	Default is "_Mirror". Sets the tile mode
	<b>_Degree</b>	(numeric)	Default is 0. Turns the Shader clockwise.
	<b>_SweepGradient</b>		
	<b>_CenterX</b>	(numeric)	Default is 0. Center point in X direction
	<b>_CenterY</b>	(numeric)	Default is 0. Center point in Y direction
	<b>_ColorArray</b>	Array of {Alpha,}Red,Green,Blue (comma delimited string) or _{Alpha,}ColorName ({comma delim.} string) or #{hn}hnhnhn (hex. string)	Default is {"_Black", "_White"}
	<b>_StopArray</b>	Array (numeric)	Default is none. Array of stop points
	<b>_Degree</b>	(numeric)	Default is 0. Turns the Shader clockwise.



Tile Modes (Source: <http://chiuki.github.io/android-shaders-filters/#/8>)

Example:

```
BUNDLE.PUT bundlePtr2, "_Shader", "_Pattern"
GR.BITMAP.LOAD sp1, "cartman.png"
GR.BITMAP.PUT bundlePtr2, "_Bitmap", sp1
BUNDLE.PUT bundlePtr2, "_Degree", 30
GR.PAINT.SET bundlePtr2
```

### Gr.paint.list <paintPointerList\_nexp>, <colorStringList\_nexp>

Returns a Paint pointer list given by colors from a list of color strings.

Note that all Paints will be new created ones, so keep care to use this command as less as possible. Other current Paint options without the color are taken over.

See Gr.color for color text definitions.

Example:

```
List.create n, paintPtrs
List.create s, colorList
FOR i = 240 TO 0 STEP -1
  List.add colorList, "_HSV" + INT$(i) % Uses the Hue color wheel
NEXT
! Return a Paint color list with the FEM (Finite element method) tension colors
! from index = 1 (blue) lowest tension, perhaps -120 N/mm²
! index = 121 (green) middle tension, perhaps 0 N/mm²
! to index = 241 (red) highest tension, perhaps 120 n/mm²
Gr.paint.get memPaint % Saves the current Paint
Gr.paint.list paintPtrs, colorList % Returns list of Paint pointers
Gr.paint.set memPaint % Load the last Paint created before
```

**Gr.text.wrap ResultArray\$[], <text\_sexp>, <widthPx\_nexp> {{, <endChecks\_sexp>}, <paint\_nexp>}**

Returns an array of strings with the result that text is broken within the boundaries defined by the width in pixels. The <endChecks\_sexp> specifies the last characters by which a line break is possible. If <endChecks\_sexp> is an empty string the breaks can be in the middle of a word. Default is "!,:. ". This command takes the current paint settings into account if <paint\_nexp> is not specified.

Example:

```
GR.OPEN "_White", 1000, -1
GR.DRAWABLE.LOAD gdo, "fly.gif"
GR.DRAWABLE.DRAW go, gdo, 100,100,900,900
GR.DRAWABLE.START go

text$ = "Hello Basic! users!\n"
text$ += "This is a text created for testing the Gr.text.wrap command.\n"
text$ += "We all hope, you have fun with all the flavors of BASIC!"
widthPx = 800

GR.COLOR "_Black"
GR.TEXT.SIZE 100

GR.TEXT.BOLD 1
BUNDLE.PUT myPaintBundle, "_TextSize", 65
BUNDLE.PUT myPaintBundle, "_TextScaleX", 1.7
BUNDLE.PUT myPaintBundle, "_Color", "_Red"
GR.PAINT.SET myPaintBundle % Overwrites the current Paint
GR.TEXT.WRAP ResultArray$[], text$, widthPx
ARRAY.LENGTH al, ResultArray$[]
BUNDLE.GET myPaintBundle, "_TextSize", lineHeight
lineSpace *= 1.1

FOR i = 1 TO al
  GR.TEXT.DRAW dp, 100, 100 + i * lineHeight, ResultArray$(i)
NEXT
GR.COLOR "_Gray", 0
GR.RECT rp, 100,100, 100 + widthPx, 100 + al * lineHeight + lineHeight / 1.1 * 0.3

GR.RENDER
DO
UNTIL 0
```



## Gr.paint.reset {<nexp>}

Force the specified Paint to default settings:

Color opaque black (255, 0, 0, 0)

Antialias ON

Style FILL (1)

Minimum stroke width (0.0)

Stroke Cap (0) → "\_Butt"

Xfermode (-1)

The parameter is optional. If the parameter is omitted or set to -1, a new current Paint is created with default settings.

**Note that all Gr.text settings will also be reseted, because these are Paint settings too.**

See also Gr.paint.set

### Gr.camera.flash <Flash\_mode\_nval>

Sets the Flash mode of the chosen camera.

The Flash\_mode numeric value specifies the flash operation:

0	Auto Flash
1	Flash On
2	Flash Off
3	Torch
4	Red-eye

The default, if no parameter is given, is (2) Flash Off.

If a failure occurs, <Flash\_mode\_nval> returns -1.

**Available if the camera is opened with Gr.open.**

If no camera is running, you can use the flash like a torch. But this is only available in Graphics Mode.

Torch Example:

```
Gr.open 255,80,80,80, 1, -1, 1
on = 3
Gr.camera.flash on
Pause 5000
off = 2
Gr.camera.flash off
```

### Gr.camera.focus <Focus\_mode\_nval>

Sets the focus mode of the chosen camera.

The Focus\_mode numeric value specifies the camera focus:

0	Auto Focus
1	Fixed Focus
2	Focus at Infinity
3	Macro Focus (close-up)
4	Continuous Picture
5	Continuous Video

The default, if no parameter is given, is Auto Focus and Continuous Picture.

If a failure occurs, <Focus\_mode\_nval> returns -1.

**Available if the camera is opened with Gr.open.**

### Gr.camera.zoom <Zoom\_factor\_nval>

Sets the zoom factor of the chosen camera.

If the zoom factor is larger as the camera's zoom maximum, <Zoom\_factor\_nval> returns the maximum zoom factor.

If a failure occurs, <Zoom\_factor\_nval> returns -1.

**Available if the camera is opened with Gr.open.**

### Gr.camera.getparam <param\_svar>

Returns the actual camera parameter.

**Can be used with Android 5+ and with Android 4 with limits.**

### Gr.camera.setparam <param\_sexp>

Sets one or more camera parameters.

**Can be used with Android 6+.**

Example:

```
GR.CAMERA.GETPARAM p$
p$ = REPLACE$(p$, ";", "\n")
PRINT p$
s$ = "effect=mono;iso=ISO200" % The ; is the delimiter.
GR.CAMERA.SETPARAM s$,1
```

### Gr.camera.directshoot <bm\_ptr\_nvar>{{,<file\_name\_sexp>} <size\_sexp>}

An image is captured as soon as the command is executed.

The optional <file\_name\_sexp> takes the wished file name.

The optional argument <size\_sexp> can be "\_Max", ("Screen") or a direct size like "1920x1080".

<bm\_ptr\_nvar> returns a bitmap. Is it smaller as the screen, it is not scaled. In the opposite case the bitmap is scaled in the manner that it only covers the screen in size, thus this is in the width or the height it is a little larger.

**Available if the camera is opened with Gr.open.**

Some Android devices are failing until now. Maybe use a Gr.camera.\*\*\*Shoot command instead.

See also File.root

### Gr.camera.select 1|2|...

Selects the Back (1) or Front(2) camera in devices with two cameras. The default camera is the back (opposite the screen) camera. [Today a device can have more than two cameras.](#)

If only one camera exists, then the default will be that camera. For example, if the device (such as the Nexus 7) only has a Front Camera then it will be the default camera. If the device does not have any installed camera apps, then there will be a run-time error message, "This device does not have a camera." In addition, a run-time error message will be shown if the device does not have the type of camera (front or back) selected.

### Gr.camera.shoot <bm\_ptr\_nvar>{, 0, 0, 0, 0, <file\_name\_sexp>}

The command calls the device's built in camera user interface to take a picture. The image is returned to BASIC! as a bitmap pointed to by the bm\_ptr numeric variable. If the camera interface does not, for some reason, take a picture, bm\_ptr will be returned with a zero value.

The command also stores the captured image into the file, "<pref base drive>/rfo-basic/data/image.jpg(png)" as default if possible. If the optional <file\_name\_sexp> is specified with "", the default file path is used. <bm\_ptr\_nvar> returns a bitmap. Is it smaller as the screen, it is not scaled. In the opposite case the bitmap is

scaled in the manner that it only covers the screen in size, thus this is in the width or the height it is a little larger.

Many of the device camera interfaces will also store the captured images somewhere else in memory with a date coded filename. These images can be found with the gallery application. BASIC! is not able to prevent these extraneous files from being created.

Note: Some devices like the Nexus 7 do not come with a built in camera interface. If you have installed an aftermarket camera application then it will be called when executing this command. You can take pictures with the Nexus 7 (or similar devices) using the other commands even if you do not have camera application installed. If the device does not have any installed camera apps, then there will be a run-time error message, "This device does not have a camera."

**Gr.camera.autoshoot <bm\_ptr\_nvar>{**<flash\_mode\_nexp>** },  
**<focus\_mode\_nexp>** , **<orientation\_nexp>** , **<take\_params\_nexp>** ,  
**<file\_name\_sexp>**}**

An image is captured as soon as the command is executed. No user interaction is required. This command can be used for untended, time-sequence image captures.

The optional flash\_mode numeric expression specifies the flash operation:

<b>0</b>	<b>Auto Flash</b>
<b>1</b>	<b>Flash On</b>
<b>2</b>	<b>Flash Off</b>
<b>3</b>	<b>Torch</b>
<b>4</b>	<b>Red-eye</b>

The default, if no parameter is given, is Auto Flash.

The optional focus\_mode numeric expression specifies the camera focus:

<b>0</b>	<b>Auto Focus</b>
<b>1</b>	<b>Fixed Focus</b>
<b>2</b>	<b>Focus at Infinity</b>
<b>3</b>	<b>Macro Focus (close-up)</b>

The default, if no parameter is given, is Auto Focus.

If you want to specify a focus mode, you must also specify a flash mode.

If the <orientation\_nexp> is not present or -1, the default orientation is Landscape.

If <take\_params\_nexp> is greater than 0 camera parameters are taken over. :

<b>0</b>	<b>No take over</b>
<b>1</b>	<b>parameters created with Gr.camera.setparam in the simple GR.open mode</b>
<b>2</b>	<b>parameters created with Gr.camera.setparam in the GR.open mode with hidden camera preview</b>

<flash\_mode\_nexp>, <focus\_mode\_nexp>, <orientation\_nexp> and picture size are overwritten by the command settings and default settings.  
Today only Android 6+!

The command also stores the captured image into the file, "<pref base drive>/rfo-basic/data/image.jpg(png)" as default. If the optional <file\_name\_sexp> is specified with "", the default file path is used. <bm\_ptr\_nvar> returns a bitmap, that only covers the screen in size, thus this is in the width or the height it is a little larger.

Keep in mind, that the time gap between two **Gr.camera.\*\*\*Shoot** commands should be round about 500 milliseconds. If a user input is necessary, a separate **Pause** command is not needed.

**Gr.camera.manualShoot** <bm\_ptr\_nvar>{, <flash\_mode\_nexp> },  
<focus\_mode\_nexp> , <orientation\_nexp> , <take\_params\_nexp> } ,  
<file\_name\_sexp>}

This command is much like **Gr.camera.autoshoot** except that a live preview is shown on the screen. The image will not be captured until the user taps the screen.

**Gr.camera.takeVideo** <file\_name\_sexp> {, <duration\_limit\_nexp> },  
<size\_limit\_nexp>}

The command calls the device's built in camera user interface to take a video.

The argument <file\_name\_sexp> sets the file name to store the video.

The file name should end with ".mp4".

If the file name is "", the file should be saved at the standard location, unfortunately, this is not always guaranteed.

The duration limit can be set in seconds with time <duration\_limit\_nexp>. This is not always guaranteed also.

The maximal file size can be set with <size\_limit\_nexp>. And the same restriction as before.

Example:

```
Gr.camera.takeVideo "video.mp4", 5, 12*1048*1048 %=12MB
```

```
File.root path$
```

```
!vv Shows the video based on an application to be selected
```

```
Browse "file://" + path$ + "/" + "video.mp4" % Absolute file path needed!
```

```
PAUSE 4000
```

```
!vv Stops playing after ~ 4 seconds in the default App for ".mp4" because "File not found".
```

```
Browse "file://" + path$ + "/" + ".mp4"
```

## Other Graphics Commands

### Gr.screen.to\_bitmap <bm\_ptr\_nvar>

The current contents of the screen will be placed into a bitmap. The pointer to the bitmap will be returned in the bm\_ptr variable. If there is not enough memory available to create the bitmap, the returned bitmap pointer is -1. Call **GETERROR\$()** for information about the failure.

Please note the idiosyncratic underscore in the command.

If a camera preview is behind the graphic screen, this preview is not visible.

So take a photo, convert (scale, crop) its size fitted to the screen, put the result in the graphic screen background, start Gr.screen.to\_bitmap and delete the background.

### Audio.info <aft\_nvar>, <bundle\_pointer\_nvar>

Returns a bundle with the system value keys:

\_Album, \_Artist, \_Title, \_BitRate, \_Duration, \_LocationPath, \_StreamMetadata,  
\_CdTrack, \_Composer, \_Compilation, \_CaptureFramerate, \_Date, \_DiscNumber  
\_StreamTitle

Streaming data are only available if an Icecast Streaming Server is called.

### Audio.load <aft\_nvar>, <filename\_sexp>|<http\_stream\_sexp>

Loads a music file **or** stream into the Audio File Table. The AFT index is returned in <aft\_nvar>. If the file **or** stream can't be loaded, the <aft\_nvar> is set to 0. Your program should test the AFT index to find out if the file or stream was loaded. If the **AFT index is 0**, you can call the **GETERROR\$()** function to get information about the error. If you use index 0 in another **Audio** command you will get a run-time error.

The file must be in the "<pref base drive>/ref-basic/data/" directories or one of its subdirectories.

You can reach outside the "<pref base drive>/ref-basic/data/" by using path fields in the filename. For example, ".././Music/Blue Danube Waltz.mp3" would access "<pref base drive>/Music/Blue Danube Waltz.mp3" **or** take a look at **File.root**.

Example:

```
FILE.ROOT dataPath$, "_Music"
fn$ = "file://" + dataPath$ + "/" + "Blue Danube Waltz.mp3"
FILE.EXISTS ok, fn$
IF ok != 0
    AUDIO.LOAD aft, fn$
    IF aft != 0
        AUDIO.STOP
        AUDIO.PLAY aft
    ENDIF
ENDIF
...
AUDIO.LOAD aft, "http://amp1.cesnet.cz:8000/cro1.ogg"
```

### Audio.play <aft\_nexp>{, <output\_nexp>}

Selects the file from the Audio File Table pointed to by <aft\_nexp> and begins to play it. There must not be an audio file already playing when this command is executed. If there is a file playing, execute audio.stop first.

Using <output\_nexp> you get control over the output channels. For the loud speakers **or** the ear phones use the 1, level-controlled by the Music mode (♪). If the ear phones **and** the loud speakers should play use the 2, level-controlled by the Alarm mode (🔔).

At this moment the current specific outer volume level is saved.

The music stops playing when the program stops running. To simply start a music file playing and keep it playing, keep the program running. This infinite loop will accomplish that:

```
Audio.load ptr, "my_music.mp3"
Audio.play ptr
Do
    Pause 5000
Until 0
```

### Audio.stop { <reset\_vol\_nexp>}

Audio.stop terminates the currently-playing music file. The command will be ignored if no file is playing. It is best to precede each Audio.play command with an Audio.stop command. If the <reset\_vol\_nexp> is set to 1 instead of 0 the outer volume levels will be reset to the saved one at Audio.play. Default <reset\_vol\_nexp> is set to 1.

### Audio.volume <left\_nexp>, <right\_nexp>{, <outer\_nexp>}

Android devices have an inner and an outer volume control. The outer one is controlled by the device's volume control buttons (key 24 and 25). Changes the inner volume of the left and right stereo channels. There must be a currently playing stream when this command is executed.

The values should range between 0.0 (lowest) to 1.0 (highest). But the human ear perceives the level of sound changes on a logarithmic scale.

The inner volume can only change in the given range of the outer volume control.

Optionally with <outer\_nexp> the outer volume level can be controlled in a range between 0.0 (lowest) to 1.0 (highest). But in this case the logarithmic scale is already taken into account. On Audio.play the current outer volume level are saved. On Audio.stop or the program stops, the outer volume level will be reset to the saved levels.

Btw. instead of "volume", "sound pressure level" <SPL> is the correct name.

~~The ear perceives a 10db change as twice as loud. A 20db change would be four times as loud.~~

~~A 1 db change would be about 0.89. One way to implement a volume control would be set up a volume table with 1db level changes. The following code creates a 16 step table.~~

```
dim volume[16]
x=1
volume [1]=x
for i = 2 to 16
    x=x*0.89
    volume [i]=x
next i
maxVolume = 10 % Wished steps
FOR vol = 0 TO maxVolume
    myVolume = (1 - (LOG(maxVolume- vol) / LOG(maxVolume)))
    !PRINT mVolume
    AUDIO.VOLUME myVolume/maxVolume, myVolume/maxVolume, 1
    ! AUDIO.VOLUME 1, 1, vol/ maxVolume
    PAUSE 1000
NEXT
```

~~Your code can select volume values from the table for use in the audio.volume command. The loudest volume would be volume[1].~~

### Audio.record.buffer <status\_nvar>, Array[]

This command is designed **only** for the PCM 16BIT format. See Audio.record.start.

No recording to a file is needed. Starting with an empty file name returns also a buffer.

The buffer is created when Audio.record.buffer is called a first time. The buffer size is depended on the sampling rate, the number of channels and the encoding type. It is settled internally in that way.

Example: For the sampling rate of 44100 Hz, 1 channel (Mono) and ~ 25 (Frames per second PCM 16BIT format) the array length is  $44100 * 1 / 25 = 1764$ .

After testing some devices, an accurate value of 25 only seems to apply for a sampling rate of 8000 Hz. A frame-rate-range from 24...25[Hz] was reached for sample rates which



are whole-number-multiples of 44100[Hz] (44100,22050,11025) at every tested device. Most -but not all- devices reached the mentioned range of frame rate even with sample rates arbitrary differing from this sequence. Example 1 can be used to determine buffer sizes for a sequence of sample rates.

The buffer is returned by the numeric array Array[].

Using two audio-channels (Audio.record.start <aC\_nexp>) instead of one simply doubles output-buffer-size, the samples are arranged individually paired.

The <status\_nvar> variable returns the status. If <status\_nvar> is 1, everything is fine. If it is 0, the PCM 16BIT format is not selected or no recording process is running. In the case of -1, the wave file recording has finished due to a file size limit and the array is now 1 in length and has a single value of -1.

If Audio.record.start <eC\_nexp> is 11 instead of 10 and the file name of Audio.record.start is empty this command is waiting for each new buffer frame.

See also [https://en.wikipedia.org/wiki/Pulse-code\\_modulation](https://en.wikipedia.org/wiki/Pulse-code_modulation)

Example 1:

```
ARRAY.LOAD sampRates[],~
48000, 44100, 40000, 32000, 24000, 22050, 16000, 11025, 8000
ARRAY.LENGTH le, sampRates[]
PRINT "-----"
PRINT "#", "sRate", "bufSize", "frameRate"
FOR i=1 TO le
  AUDIO.RECORD.START "" , 1, 3, -1, sampRates[i], -1, 1
  AUDIO.RECORD.BUFFER id, buf[]
  AUDIO.RECORD.STOP
  ARRAY.LENGTH bufSz, buf[]
  PRINT INT$(i), INT$(sampRates[i]), ";";
  PRINT INT$(bufsz); ";";
  PRINT STR$(ROUND(sampRates[i]/bufSz,2))
NEXT
CONSOLE.SAVE "bufferSizes.txt"
END
```

### **Audio.record.peak <level\_nvar>**

With this PEAK command you get with <level\_nvar> the raw max. sound pressure level amplitude value between to calls. If the PCM 16BIT mode is running, only the last buffer will be evaluated.

The Audio.record.START command initializes the PEAK process for the first time.

If no current recording process available, PEAK returns -1.

**Audio.record.start** <fn\_sexp>{,,,,,,,,, <so\_nexp>, <oF\_nexp>, <eC\_nexp>, <sR\_nexp>, <eBR\_nexp>, <aC\_nexp>, <mFS\_nexp>, <lat\_nexp>, <lon\_nexp>}

Start audio recording using the microphone as the audio source. The recording will be saved to the specified file. Recording will continue until the **Audio.record.stop** command is issued.

**Command OPTIONS:**

Argument	Description	Defaults
<fn_sexp>	File Name If <fn_sexp> is an empty String nothing is saved.	
<so_nexp>	Source 1* MICROPHONE; 5 CAMCORDER; 6 VOICE_RECOGNITION; 7 VOICE_COMMUNICATION	1
<oF_nexp>	Output Format 1* THREE_GPP(.g3p); 2 MPEG_4 (.mp4, .m4a) 3 WAVE_16BIT (.wav) and switches autom. to Encoder 10 (PCM_16BIT)	1

<eC_nexp>	Encoder 1* AMR_NB 2 AMR_WB 3 AAC 4 HE_AAC 5 AAC_ELD 6 Ogg Vorbis (API level 21) 10 PCM_16BIT (autom. switching to Output Format WAVE_16BIT (.wav) ) 11 PCM_16BIT <b>Waiting</b> for each new buffer frame, if the file name <b>is empty</b> . Nothing is saved.	1
<sR_nexp>	Sampling Rate in samples per second 44100Hz is currently the only rate that is guaranteed to work on all devices, but other rates such as 5.5125, 6.615, 8, 9.6, <b>11.025</b> , <b>16</b> , 18.9, <b>22.05</b> , 27.42857, 32, 33.075, 37.8, 44.1, 48 kHz. may work on some devices.	44100
<eBR_nexp>	Encoding Bit Rate in bits per second Ignored by PCM_16BIT	96000
<aC_nexp>	Audio Channels 1 or 2* For one channel you get a mono output in a stereo file if not a PCM 16 BIT encoder is choosed!	2
<mFS_nexp>	Maximal File Size in bytes, but the real size is more less! If the max. file size is reached, the recording stops sooner or later!	-1
<lat_nexp>	Latitude -90 to 90 degree Ignored by PCM_16BIT	
<lon_nexp>	Longitude -180 to 180 degree Ignored by PCM_16BIT	

#### Attention:

Some results are device, sampling rate, encoding bit rate and/or codec depended.

#### Example:

```
REM Start of BASIC! Program audio_recording.bas
Device dbp
Bundle.get dbp, "OS", myOs$
Print myOs$
```

```

DEBUG.ON
filename$ = "NewSound.g3p"
filename$ = "NewSound.m4a"
! file.delete done, filename$ %If you use the RFO-BASIC you have to delete first!!!
! Command OPTIONS:
so = 1 %Source 0 Default; 1* Mic; 5 CAMCORDER; 6 VOICE_RECOGNITION;~
! (7 VOICE_COMMUNICATION)
oF = 2 %OutputFormat 1* THREE_GPP(.g3p); 2 MPEG_4 (.mp4, .m4a)
eC = 3 %Encoder 1* AMR_NB; (2 AMR_WB; 3 AAC); (4 HE_AAC; 5 AAC_ELD)
sR = 48000 %(SamplingRate in samples per second) 44100*
eBR = 80000 %(EncodingBitRate in bits per second) 96000*
aC = 2 %(AudioChannels 1 or 2*) For one channel you get a mono output in a stereo file!
mFS = 280000 %MaxFileSize in bytes, but the real size is more less! -1*
! If the max. file size is reached, the recording stops sooner or later!
lat = 50 %(Latitude -90 to 90 degree)
lon = 150 %(Longitude -180 to 180 degree)

AUDIO.RECORD.START filename$,so,oF,eC,sR,eBR,aC,mFS,lat,lon
FOR i = 1 TO 20
  PAUSE 1000
  ! With this PEAK command you get the max. amplitude between to calls.
  ! The START command init the PEAK process for the first time.
  ! If no current recording process available, PEAK returns -1.
  AUDIO.RECORD.PEAK m
  FILE.SIZE fS, filename$
  PRINT "Peak "; m; " File size "; fS
NEXT i

FILE.SIZE fS, filename$
PRINT " Sound file size in byte: ";fS
AUDIO.RECORD.STOP
AUDIO.LOAD ptr, filename$
AUDIO.PLAY ptr
DO
UNTIL 0
END

```

See also:

<https://developer.android.com/guide/topics/media/media-formats.html>

### STT.listen {{<prompt\_sexp>}, <extras\_bundle\_nexp>}

Start the voice recognize process by displaying a "Speak Now" dialog box. The optional prompt string expression <prompt\_sexp> sets the dialog box's prompt. If you do not provide the prompt parameter, the default prompt "BASIC! Speech To Text" is used. Begin speaking when the dialog box appears.

The recognition will stop when there is a pause in the speaking.

**STT.results** should be executed next.

Note: **STT.listen** is not to be used in HTML mode.

The extras bundle <extras\_bundle\_nexp> controls more options:

Key	Value	
<b>_Hidden</b>	0* or 1 (numeric)	Opens the recognizer without any dialogues. If an error occurs only "Recognition Canceled" will be returned. This option takes only effect within Console Mode. Default is 0. In this case the recognizer is in front.
<b>_Off</b>	0* or 1 (numeric)	The Values are set, but a recognition will be not performed. An option in case of using GW-Lib within HTML Mode. Default is 0. In this case the recognizer will be performed.
<b>_MaxResults</b>	numeric	Optional limit on the maximum number of results to return. If omitted the recognizer will choose how many results to return.
<b>_Language</b>	String	Optional IETF language tag, for example "en-US". This tag informs the recognizer.
<b>_MinimumLength</b>	numeric	The minimum length of an utterance. We will not stop recording before this amount of time. Note that it is extremely rare you'd want to specify this value. If you don't have a very good reason to change these, you should leave them as they are per default. Note also that certain values may cause undesired or unexpected results - use judiciously! Additionally, depending on the recognizer implementation, these values may have no effect.

<b>_CompletSilence</b>	numeric	The amount of time that it should take after we stop hearing speech to consider the input complete. Note that it is extremely rare you'd want to specify this value. If you don't have a very good reason to change these, you should leave them as they are per default. Note also that certain values may cause undesired or unexpected results - use judiciously! Additionally, depending on the recognizer implementation, these values may have no effect.
<b>_PossiblyCompleteSilence</b>	numeric	The amount of time that it should take after we stop hearing speech to consider the input possibly complete. This is used to prevent the end pointer cutting off during very short mid-speech pauses. Note that it is extremely rare you'd want to specify this value. If you don't have a very good reason to change these, you should leave them as they are per default. Note also that certain values may cause undesired or unexpected results - use judiciously! Additionally, depending on the recognizer implementation, these values may have no effect.
<b>_WebSearch</b>	0* 1 Web search and others 2 Only web search (numeric) [Does not work properly with Google Assistant]	<del>Min. Jelly Bean 4.1 (API 16)</del> Prompts the user for speech, send it through a speech recognizer, and either display a web search result or trigger another type of action based on the user's speech. For security reasons the mode falls back to normal recognition, if the screen is off or the device is locked. If Google Assistant installed, it will be used. The problem is, Google Assistant can not be closed by voice.

<b>_HandsFree</b>	0* or 1 (numeric)	<del>Min. Jelly Bean 4.1 (API 16)</del> Launches a start of a HandsFreeApp in a mode that will prompt the user for speech without requiring the user's visual attention or touch input. This activity may be launched while device is locked in a secure mode. Special care is be taken to ensure that the voice actions that are performed while hands free cannot compromise the device's security. The called application can return a list, a String or a Number from type Double in the returned intent <b>extra</b> bundle. But your result is in each case only a list from type Sting. If you create with Bundle.out your result, you can use <b>only</b> a string or a number.
<b>_Package</b>	String	<del>Min. Jelly Bean 4.1 (API 16)</del> Only in conjunction with _HandsFree. Package name of the whished App. Example: "com.rfo.basicOli"
<b>_Component</b>	String	<del>Min. Jelly Bean 4.1 (API 16)</del> Only in conjunction with _HandsFree and _Package. Example: "com.rfo.basicOli.Basic"
<b>_Data</b>	String as URL	<del>Min. Jelly Bean 4.1 (API 16)</del> Only in conjunction with _HandsFree and _Package. Example: FILE.ROOT fp\$, "_External" filePath\$ = " <u>file:///</u> " + fp\$ + "rfo-basic/source" mUrl\$ = filePath\$ + "/" + "freeHandsDemo.bas"

If the following paragraph in the AndroidManifest.xml is accessible, the program is also a possible hands free receiver like Google Assistant. But the advantage is, the program can be closed by voice. If you create a **normal** APK from your program, the paragraph should be deleted.

```
<!-- vv 2017-11-25gt Comment it out if it should not be a hands free receiver, too. -->
<intent-filter >
    <action android:name="android.speech.action.VOICE_SEARCH_HANDS_FREE" />
    <category android:name="android.intent.category.DEFAULT" />
</intent-filter>
<!-- ^^ 2017-11-25gt -->
```

### **TTS.kill**

TTS.kill stops the TTS process without waiting for finish. With this command you have to code carefully! You have to be guaranteed, that all following commands including until TTS.STOP will not be executed. Following **TTS.kill**, if you want to run **TTS.speak** or **TTS.speak.toFile** again, you will have to run **TTS.init** again.



## Information About Your Android Device

### Device Command Overview

You can get information about your Android device with the **Device** command:

- The Device Brand, Device Model, Device Type, and OS
- The Language and Locale
- The PhoneType, PhoneNumber, and DeviceID
- The SN, MCC/MNC, and Network Provider stored on the SIM, if there is one.

The **Device** command has two forms that differ only in the type of the parameter, which determines the format of the returned data. Both forms return the same information, as shown in this table:

Key	Values	Meaning	Example (from emulator)
<b>Brand</b>	Any string	Brand name assigned by device manufacturer	generic
<b>Model</b>	Any string	Model identifier assigned by device manufacturer	sdk
<b>Device</b>	Any string	Device identifier assigned by device manufacturer	generic
<b>Product</b>	Any string	Product identifier assigned by device manufacturer	sdk
<b>OS</b>	OS Version	Android operating system version number	4.1.2
<b>Language</b>	Language name	Default language of this device	English
<b>Locale</b>	Locale code	Default locale code, typically language and country	en_US
<b>PhoneType</b>	<b>GSM, CDMA, SIP, or None</b> or <b>Not available</b>	Type of phone radio in this device	GSM
<b>PhoneNumber</b>	String of digits or <b>Not available</b>	Phone number registered to this device, if any	15555215554
<b>DeviceID</b>	String of digits or <b>Not available</b>	The unique device ID, such as the IMEI <b>until Android 8.1</b>	0000000000000000
<b>SIM SN</b>	String of digits or <b>Not available</b>	Serial number of the SIM card, if one is present and it is accessible	890141032111185 10720
<b>SIM MCC/MNC</b>	String of digits or <b>Not available</b>	The "numeric name" of the provider of the SIM, if present and accessible	310260
<b>SIM Provider</b>	Name string or <b>Not available</b>	The name of the provider of the SIM, if present and accessible	Android

The last six items access your device's telephony system and SIM card. If your device has no telephone, or BASIC! does not have permission to access them, the fields are set to neutral values: "None", "Not available", or a string of "0" characters.

In addition, there are convenience commands to retrieve only the Locale or the Language. The information returned by Device is static. To get dynamic information, use the Phone.Info command.

### Device <svar>

Returns information about your Android device in the string variable <svar>. For this command you need **Phone permissions**. Each item has this form:

key = value

The names key and value refer to the first two columns of the table in the **Device** overview above. The items are placed in a single string, separated by newline characters.

Formatted this way, if you **Print** the string it is displayed with one item on each line. You can separate the individual items with **Split**:

DEVICE info\$	% all info in one string
SPLIT info\$[], info\$, "\n"	% each array element is one item, "<k> = <v>"
SPLIT lang_line\$[], info\$[6], " = "	% split the Language item, two-element array
lang\$ = lang_line\$[2]	% lang\$ is the language name, like "English"

### Device <nvar>

Returns information about your Android device in a Bundle. If you provide a variable that is not a valid Bundle pointer, the command creates a new Bundle and returns the Bundle pointer in your variable. Otherwise it writes into the Bundle your variable or expression points to. For this command you need **Phone permissions**.

The Bundle keys are shown in the first column of the table in the **Device** overview above.

DEVICE info	% all info in a Bundle
BUNDLE.GET info, "Language", lang\$	% lang\$ is the language name, like "English"

### Device.os <api\_nvar>{<release\_svar>, <codename\_svar>, <incremental\_svar>, <security\_svar>, <base\_os\_svar>}

Returns the os as a number. For this command you need **no Phone permissions**.

Options:

- <release\_svar> returns  
the user-visible version string. E.g., "1.0" or "3.4b5".
- <codename\_svar> returns  
the current development codename, or the string "REL" if this is a release build.
- <incremental\_svar> returns  
the internal value used by the underlying source control to represent this build.  
E.g., a perforce changelist number or a git hash.
- <security\_svar> returns  
the user-visible security patch level. (Needs API level 23.)
- <base\_os\_svar> returns  
the base OS build the product is based on. (Needs API level 23.)

### Device.USB <bundlePointer\_nvar>

Returns the parameters of plugged in USB devices.

Following keys are supported "\_Deviceld", "\_VendorId", "\_ProductId", "\_DeviceClass", "\_InterfaceClass0", "\_DeviceSubclassId", "\_DeviceName", "\_SerialNumber", "\_ManufacturerName", "\_ProductName", "\_UsbInterface", "\_Driver" and "\_AllAsString". The keys "\_SerialNumber", "\_ManufacturerName" and "\_ProductName" are available on Android 5+.

Starting on Android 10+ a permission has to be granted for each USB device. Is a dialog canceled or after five seconds not confirmed the property enumeration is skipped.

Example:

```
DEVICE.USB bp
BUNDLE.KEYS bp, lp
LIST.SIZE lp, n
FOR i = 1 TO n
    BUNDLE.GB bp, int$(i), pUSB
    BUNDLE.GET pUSB, "_AllAsString", aAS$
    PRINT aAS$
NEXT
```

#### **Device.get.brightness <brigh\_nvar>**

Returns the brightness in the range from 0 to 255. Needs WRITE\_SETTINGS permission.

#### **Device.set.brightness <brigh\_nexp>**

Sets brightness mode to manual and sets the brightness in the range from 0 to 255. Needs WRITE\_SETTINGS permission.

#### **Device.auto.brightness <brigh\_nexp>**

Sets brightness mode to automatic. Needs WRITE\_SETTINGS permission. App.settings {<package\_sexp>}

Calls the application settings of the app with the package-id <package\_sexp>. If the optional package-id is not given, application settings of the running Basic engine or the settings of the executed application created with Basic! is opened.

## Device Settings

<https://developer.android.com/reference/android/provider/Settings#public-constructors>

Example:

```
FN.DEF ViewSetting(setting$)
  LIST.CREATE S, commandListPointer
  LIST.ADD commandListPointer~
  "new Intent(" + CHR$(34) + setting$ + CHR$(34) + ");" ~
  "addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);" ~
  "addFlags(Intent.FLAG_ACTIVITY_CLEAR_WHEN_TASK_RESET);" ~
  "EOCL"
  LIST.CREATE S, resultListPointer
  LIST.ADD resultListPointer~
  "EORL"
  BUNDLE.PL appVarPointer,"_CommandList",commandListPointer
  ! A Result List is needed if you want to waituntil the dialog is finished.
  BUNDLE.PL appVarPointer,"_ResultList",resultListPointer

APP.SAR appVarPointer
  List.kill.last : List.kill.last : Bundle.kill.last % Clean up Lists and Bundles
FN.END

! From
! https://developer.android.com/reference/android/provider/Settings#public-constructors
! We get an action like ACTION_BLUETOOTH_SETTINGS
! we have to convert into android.settings.BLUETOOTH_SETTINGS

actionCode$ = "ACTION_AIRPLANE_MODE_SETTINGS"
setting$ = "android.settings." + REPLACE$(actionCode$, "ACTION_", "")
Bundle.clear myVarPointer
myVarPointer = ViewSetting (myVarPointer, setting$)

actionCode$ = "ACTION_BLUETOOTH_SETTINGS"
setting$ = "android.settings." + REPLACE$(actionCode$, "ACTION_", "")
Call ViewSetting (setting$)

actionCode$ = "ACTION_PRIVACY_SETTINGS"
setting$ = "android.settings." + REPLACE$(actionCode$, "ACTION_", "")
app.start setting$
Dialog.message "Dialog Finished", "", sel, "Ok" % Set a dialog box to wait.
PRINT ok
do
until 0
END
```

## HTML Commands

### Html.open {{{<Title\_nexp>}, <Orientation\_nexp>}, <securityLevel\_nexp>}

This command must be executed before using the HTML interface.

The Statusbar will be shown on the Web Screen if the <Title\_nexp> is 1 ~~true (not zero)~~. If the <Title\_nexp> is not present, the Status bar will not be shown.

If <Title\_nexp> is **negative**, Html.open expects a layout bundle defined by items of the following table.

The orientation upon opening the HTML screen will be determined by the <Orientation\_nexp> value. <Orientation\_nexp> values are the same as values for the Html.orientation command (see below). If the <Orientation\_nexp> is not present, the default orientation is determined by the orientation of the device.

Both <Title\_nexp> and <Orientation\_nexp> are optional; however, a <Title\_nexp> must be present in order to specify an <Orientation\_nexp>.

Executing a second HTML.OPEN before executing HTML.CLOSE will generate a run-time error.

A security level can be set by <securityLevel\_nexp>:

>= 4 Javascript disabled

< 4 Javascript enabled

< 3 JavaScript can open windows automatically


< 2 Set allow file access from file URLs

< 1 Set allow universal access from file URLs

Default is 3.

The optional options bundle (-) <Title\_nexp> controls the layouts of the Action and Navigation bars:

Table of Layout Control Options		
Key	Value	Description
<b>_ShowActionbar</b>	0 or 1 (numeric)	If 1 Show the Action bar if it is not currently showing. It is needed to show titles and to change the background color of the Statusbar. If 0 (default) Hide the Actionbar if it is currently activated.
<b>_Title</b>	String by default	Set the action bar's title.
<b>_Subtitle</b>	String by default	Set the action bar's subtitle.
<b>_TitleShow</b>	0 or 1 (numeric)	If 1 (default) Show the Action bar if it is not currently showing. It will resize application content to fit the new space available. If 0 Hide the Actionbar if it is currently showing. It will resize application content to fit the new space available.

<b>_TitleIcon</b>	Icon file path	Add a large icon to the notification content view. <a href="http://romannurik.github.io/AndroidAssetStudio/index.html">http://romannurik.github.io/AndroidAssetStudio/index.html</a>
<b>_TitleHomeEnabled</b>	0 or 1 (numeric)	Set whether to include the application home accordance in the action bar. Home is presented as an activity icon. Have to be 1 if you want to show the icon. Have to be 0 if you want to hide the icon. The default setting is API dependent.
<b>_TitleBackground</b>	Background file path	
<b>_TitleHtml</b>	0 or 1 (numeric)	Returns displayable styled text from the provided HTML string. But not all tags are supported.  Uses parts of TagSoup library to handle real HTML, including all of the brokenness found in the wild. <b> <big> <font size="..." color="..." face="..."> <h1>, <h2>, <h3>, <h4>, <h5>, <h6> <i> <small> <strike>? < A.7 <strong> <sub> <sup> <tt>? <u>  <b>Replace Space with &amp;#160, &amp; with &amp;amp, &lt; with &amp;lt, &gt; with &amp;gt, " with &amp;quot if necessary.</b> Usable for Title and Subtitle. Keep in mind that the ActionBar height will not be expanded.

<b>_ShowStatusbar</b>	0, 1 or 2 (numeric)	<p>If 1 (default) The Status bar will be displayed.</p> <p>If 2 The Status bar will be transparent displayed.</p> <p>Min. Lollipop 5.0 (API 21)</p> <p>If 0 The Status bar will be hidden to the background.</p> <p>Min. Nougat 7.0 (API 24) Will be switched to option 2 or 1 if the current API level is lower.</p>
<b>_StatusbarColor</b>	{Alpha,}Red,Green,Blue (comma delimited string) or _{Alpha,}ColorName (comma delim.} string) or #{hn}hnhnhn (hex. string)	<p>Min. Lollipop 5.0 (API 21)</p> <p>Note, the ActionBar has to be activated by <b>_ShowActionBar</b>.</p>
<b>_StatusbarLight</b>	0 or 1 (numeric)	<p>If 0 (default) The Status bar background is dark. In this case the <b>bar content</b> will be <b>light</b>.</p> <p>If 1 The Status bar background is light. In this case the <b>bar content</b> will be <b>dark</b>.</p> <p>Min. Lollipop 5.0 (API 21)</p>
<b>_ShowNavigationbar</b>	0, 1 or 2 (numeric)	<p>If 1 (default) The Navigation bar will be displayed.</p> <p>If 2 The Navigation bar will be transparent displayed.</p> <p>Min. Lollipop 5.0 (API 21)</p> <p>If 0 The Navigation bar will be hidden to the background.</p> <p>Min. Nougat 7.0 (API 24) Will be switched to option 2 or 1 if the current API level is lower.</p>
<b>_NavigationbarColor</b>	{Alpha,}Red,Green,Blue (comma delimited string) or _{Alpha,}ColorName (comma delim.} string) or #{hn}hnhnhn (hex. string)	<p>Min. Lollipop 5.0 (API 21)</p>

<b>_NavigationBarLight</b>	0 or 1 (numeric)	<p>If 0 (default) The Navigation bar background is dark. In this case the <b>bar content</b> will be <b>light</b>.</p> <p>If 1 The Navigation bar background is light. In this case the <b>bar content</b> will be <b>dark</b>.</p> <p>Min. Lollipop 5.0 (API 21)</p>
<b>_Menu</b>	Menu Bundle Pointer	<p>Creates menu entries. A successful selection will be returned as a human readable JSON string. See the example at Console.title for more details.</p>

See also: [GR.open](#), [Console.title](#), [Select](#)



### Html.get.datalink <data\_svar>

A datalink provides a method for sending a message from an HTML program to the BASIC! programmer. There are two parts to a datalink in an HTML file:

1. The JavaScript that defines the datalink function
2. The HTML code that calls the datalink function.

The BASIC! Program requires a mechanism for communicating with a website's HTML code.

**Html.get.datalink** gets the next datalink string from the datalink buffer. If there is no data available then the returned data will be an empty string (""). You should program a loop waiting for data:

```
DO
  HTML.GET.DATALINK data$
UNTIL data$ <> ""
```

The returned data string will always start with a specific set of four characters—three alphabetic characters followed by a colon (":"). These four characters identify the return datalink data type. Most of the type codes are followed by some sort of data. The codes are:

**BAK:** The user has tapped the BACK key. The data is either "1" or "0".

If the data is "0" then the user tapped BACK in the start screen. Going back is not possible therefore HTML has been closed.

If the data is "1" then going back is possible. The BASIC! programmer should issue the command **Html.go.back** if going back is desired.

**LNK:** The user has tapped a hyperlink. The linked-to url is returned. The transfer to the new url has not been done. The BASIC! programmer must execute an **Html.load.url** with the returned url (or some other url) for a transfer to occur.

**ERR:** Some sort of fatal error has occurred. The error condition will be returned. This error code always closes the html engine. The BASIC! output console will be displayed.

**FOR:** The user has tapped the Submit button on a form with action = "FORM". The form name/value pairs are returned.

**DNL:** The user has clicked a link that requires a download. The download url is supplied. It is up to the BASIC! programmer to do the download.

**DAT:** The user has performed some action that has caused some JavaScript code to send data to BASIC! by means of the datalink. The JavaScript function for sending the data is:

```
<script type="text/javascript">
  function doDataLink(data) {
    Android.dataLink(data);
  }
</script>
```

**STT:** Calls the speech recognition. Note, for detection the first four characters are used.

**EJS:** Returns the result of **Html.evaluate.js** if any.

**CME:** Returns a Console Message as String from type

CME:<lineNumber>,<sourceId>,<messageLevel>,<message>

### Html.evaluate.js <js\_sexp>

Sends a JavaScript script to the current HTML page or JavaScript if loaded. This command waits until the HTML page returns a result from the sender script. A complete function script can also be used without a loaded script. The result will be posted via Html.get.datalink with the header "EJS:".

Also an option to call a function is HTML.LOAD.URL "javascript:text('BASIC! for ever!')" used in data/htmldemo1.html.

Example:

```
closeHtml = 0
Html.open 0, -1
Html.load.string "<script>function callJS(str) { return str; } <\script>"
Html.evaluate.js "callJS( 'BASIC! for ever!' )"
DO
  PAUSE 100
UNTIL closeHtml
Html.close
OnHtmlReturn:
  Html.get.datalink data$
  IF IS_IN("EJS:", data$) = 1 THEN PRINT MID$(data$, 5) : closeHtml = 1
Html.onHtmlReturn.resume
```

### Html.screenshot <filename\_sexp>

Saves the current screen to a file. The default path is "<pref base drive>/rfo-basic/data/". The file will be saved as a JPEG file if the filename ends in ".jpg".

The file will be saved as a PNG file if the filename ends in anything else (including ".png").

### Html.to.pdf <filename\_sexp> {{{, <paperformat\_sexp>}, <orientation\_sexp>}, <resolution\_nexp>}, <color\_sexp>}

Saves the current html content to a file. The default path is "<pref base drive>/rfo-basic/data/".

The chosen paper format will be set by <paperformat\_sexp> default is "ISO\_A4".

The optional <orientation\_sexp> defines the sheet orientation. "L" is landscape and "P" the default portrait. The resolution can be set by <resolution\_nexp> in dots per inch. The default setting is 600 dpi. By default <color\_sexp> is set to "C" for colored output. "M" saves the content monochrome.

**Note that to override, you have to delete the old PDF file first.**

Example:

```
HTML.OPEN 0, -1
!html.load.url "htmlDemo1.html"
mM$ = "Hello<br>World"
HTML.LOAD.STRING mM$
HTML.PAPERFORMATS pfb
BUNDLE.KEYS pfb, pfl
Dialog.select sel, pfl, "Choose a paper format"
List.get pfl, sel, pf$
fileName$ = "html.pdf"
FILE.EXISTS fOk, fileName$
```

```
IF fOk THEN FILE.DELETE dOk, fileName$
HTML.TO.PDF fileName$, pf$ , "P"
DO % Needed, because asynchronous background task.
  PAUSE 100
  FILE.EXISTS fOk, fileName$
UNTIL fOk
HTML.CLOSE
```

### **Html.paperformats <bundlePointer\_nexp>**

Returns possible paper formats as a bundle. The bundle key is also the paper format key name. The bundle entry returns more information about the specified format as a delimited string.

### **OnHtmlReturn:**

Interrupt label that traps if HTML data arrived. BASIC! executes the statements following the **OnHtmlReturn:** label until it reaches a **Html.onHtmlReturn.resume**.

See also `Html.get.datalink`, `MenuItem.get.datalink`

### **Html.onHtmlReturn.resume**

Resumes execution at the point in the BASIC! program where the **OnHtmlReturn:** interrupt occurred.

### **IS\_HTML()**

IS\_HTML returns the HTML mode status. If it is 1 the HTML mode is enabled. Is 0 returned the HTML mode is not open.

See also `Console.Save`

## BACKGROUND()

A running BASIC! program continues to run when the HOME key is tapped. This is called running in the Background. When not in the Background mode, BASIC! is in the Foreground mode. BASIC! exits the Background mode and enters the Foreground mode when the BASIC! icon on the home screen is tapped.

Sometimes a BASIC! programmer wants to know if the program is running in the Background. One reason for this might be to stop music playing while in the Background mode.

The **BACKGROUND()** function returns true (1) if the program is running in the background. It returns false (0) if the program is not running in the background.

It returns (3) if the display screen is **off**.

~~Min. Jelly Bean 4.1 (API 16):~~

It returns (4) if the device is **locked**.

It returns (5) if the display screen is **on** and the device is **locked**.

If you want to be able to detect Background mode while Graphics is open, you must not call **Gr.render** while in the Background mode. Doing so will cause the program to stop running until the Foreground mode is re-entered. Use the following code line for all **Gr.render** commands:

```
IF !BACKGROUND() THEN GR.RENDER
```

## HYPOT(<nexp\_x>, <nexp\_y>)

Returns  $\text{SQR}(x^2+y^2)$  without intermediate overflow or underflow.

## ATAN(<nexp>)

Returns the arc tangent of the angle <nexp>, in the range of  $-\pi/2$  through  $\pi/2$ . The units of the angle are radians.

Returns the closest double approximation of the arc tangent of the argument within the range  $[-\pi/2..\pi/2]$ .

Special cases:

- $\text{atan}(+0.0) = +0.0$
- $\text{atan}(-0.0) = -0.0$
- $\text{atan}(+\text{infinity}) = +\pi/2$
- $\text{atan}(-\text{infinity}) = -\pi/2$
- $\text{atan}(\text{NaN}) = \text{NaN}$

## ATAN2(<nexp\_y>, <nexp\_x>)

Returns the angle *theta* from the conversion of rectangular coordinates (x, y) to polar coordinates (r, *theta*), in the range of  $-\pi$  through  $\pi$ . (Please **note the order** of the parameters in this function.)

Special cases:

- $\text{atan2}(\text{anything}, \text{NaN}) = \text{NaN}$ ;
- $\text{atan2}(\text{NaN}, \text{anything}) = \text{NaN}$ ;
- $\text{atan2}(+0.0, \text{anything but NaN}) = +0.0$
- $\text{atan2}(-0.0, \text{anything but NaN}) = -0.0$
- $\text{atan2}(+0.0, \text{anything but NaN}) = +\pi$
- $\text{atan2}(-0.0, \text{anything but NaN}) = -\pi$
- $\text{atan2}(\text{anything but 0 and NaN}, 0) = +\pi/2$
- $\text{atan2}(\text{anything but 0 and NaN}, 0) = -\pi/2$
- $\text{atan2}(\text{anything but infinity and NaN}, +\text{infinity}) = +0.0$
- $\text{atan2}(\text{anything but infinity and NaN}, +\text{infinity}) = -0.0$
- $\text{atan2}(\text{anything but infinity and NaN}, -\text{infinity}) = +\pi$
- $\text{atan2}(\text{anything but infinity and NaN}, -\text{infinity}) = -\pi$
- $\text{atan2}(+\text{infinity}, +\text{infinity}) = +\pi/4$       **+ 45° 1. Quadrant**
- $\text{atan2}(-\text{infinity}, +\text{infinity}) = -\pi/4$       **- 45° 4. Quadrant**
- $\text{atan2}(+\text{infinity}, -\text{infinity}) = +3\pi/4$       **+ 135° 2. Quadrant**
- $\text{atan2}(-\text{infinity}, -\text{infinity}) = -3\pi/4$       **- 135° 3. Quadrant**
- $\text{atan2}(+\text{infinity}, \text{anything but 0, NaN, and infinity}) = +\pi/2$
- $\text{atan2}(-\text{infinity}, \text{anything but 0, NaN, and infinity}) = -\pi/2$

Example:

```
FN.DEF ATAN4(y, x) % It returns the radiant for the term y / x in the range [0 ... 2*PI]
  result = ATAN2( y, x)
  IF result < 0 THEN result = PI()*2 + result
  IF result = PI()*2 THEN result = 0
  FN.RTN result
FN.END
PRINT ATAN4(-1, -1), TODEGREES(ATAN4(-1, -1))
```

## CLAMP(<value\_nexp>, <min\_nexp>, <max\_nexp>)

This method takes the numerical value <value\_nexp> and ensures it fits in a given numerical range. If the number is smaller than the minimum <min\_nexp> required by the range, then the minimum <min\_nexp> of the range will be returned. If the number is higher than the maximum <max\_nexp> allowed by the range then the maximum <max\_nexp> of the range will be returned.

It is also a one liner, max(mMin, min(mMax, mVal))

If you have a lot of values use List.join, like the line above. It is much faster.

Example:

```
x = 20
grid_size = 100
grid_width = 1000
gx = CLAMP(x/grid_size, 1, grid_width)
! gx = MAX(mMin, MIN(mMax, mVal))
gx = MAX(1, MIN(grid_width, x/grid_size))
LIST.CREATE n, mListX
LIST.ADD mListX, x
LIST.ADD mListX, 40, 60, 80, 100, 120
LIST.CREATE n, mListR1
LIST.JOIN mListR1, mListX, STR$(grid_size) , "", "_/8"
! DEBUG.DUMP.LIST mListR1
LIST.CREATE n, mListR2
LIST.JOIN mListR2, STR$(grid_width), mListR1, "", "_min"
! DEBUG.DUMP.LIST mListR2
LIST.CREATE n, mListR3
LIST.JOIN mListR3, STR$(1), mListR2, "", "_max"
DEBUG.DUMP.LIST mListR3
```

See also [List.Join](#), [Array.Math](#)

## **EVEN(<nexp>)**

Returns 1 if the integer part of <nexp> is even. Otherwise 0 is returned.

If <nexp> is NaN or infinite 0 is returned also.

Maybe you have to round for cases like 79.999999999 before.

## **ODD(<nexp>)**

Returns 1 if the integer part of <nexp> is odd. Otherwise 0 is returned.

If <nexp> is NaN or infinite 0 is returned also.

Maybe you have to round for cases like 80.999999999 before.

## **CLOCK({<nano\_nexpr>})**

Returns the time in milliseconds since the last start and include deep sleep. This clock is guaranteed to be monotonic, and continues to tick even when the CPU is in power saving modes, so is the recommend basis for general purpose interval timing. If the optional <nano\_nexpr> is greater than 0, nanoseconds are returned. The default is 0.

To compare two nano Time values

```
tic1 = Clock(1)
```

```
...
```

```
tic2 = Clock(1)
```

One should use `tic2 - tic1 < 0`, not `tic2 < tic1`, because of the possibility of numerical overflow.

Note, that one day has 86,400,000,000,000 nanoseconds. That are 14 digits. The double values have only 15 significant digits.

BigD.nanoTime is also an option, but has a different time base.

### USING\$(<locale\_sexp> , <format\_sexp> { , <exp>}...)

Returns a string, using the locale and format expressions to format the expression list. This function gives BASIC! programs access to the `Formatter` class of the Android platform. You can find full documentation here:

<https://developer.android.com/reference/java/util/Formatter.html>.

The <locale\_sexp> is a string that tells the formatter to use the formatting conventions of a specific language and region or country. For example, "en\_US" specifies American English conventions.

The <format\_sexp> is a string that contains *format specifiers*, like "%d" or "%7.2f," that tell the formatter what to do with the expressions that follow.

The format string is followed by a list of zero or more expressions. Most format specifiers take one argument from the list, in order. If you don't provide as many arguments as your format string needs, you will get a detailed Java error message.

Each expression must also match the type of the corresponding format specifier. If you try to apply a string format specifier, like "%-7s", to a number, or a floating point specifier, like "%5.2f" to a string, you will get a Java error message.

### Locale expression

The **USING\$()** function can localize the output string based on language and region. The locale specifies the language and region with standardized codes. The <locale\_sexp> is a string containing zero or more codes separated by underscores.

The function accepts up to three codes. The first must be a language code, such as "en", "de", or "ja". The second must be a region or country code, such as "FR", "US", or "IN". Some language and country combinations can accept a third code, called the "variant code".

The function also accepts the standard three-letter codes and numeric codes for country or region. For example, "fr\_FR", "fr\_FRA", and "fr\_250" are all equivalent.

If you want to use the default locale of your Android device, make the <locale\_exp> an empty string (""), or leave it out altogether. If you leave it out, you must keep the comma:

**USING\$( "", x)**

If you make a mistake in the <locale\_sexp>, you may get an obscure Java error message, but more likely your locale will be ignored, and your string will be formatted using the default locale of your device.

Android devices do not support all possible locales. If you specify a valid locale that your device does not understand, your string will be formatted using the default locale.

### Format expression

If you are familiar with the *printf* functions of C/C++, Perl, or other languages, you will recognize most of format specifiers of this function. The format expression is exactly the same as format string of the Java *Formatter* class, or the *format(String, Object...)* method of the Java *String*, with two exceptions: Boolean format specifiers are not supported, and hex hash specifiers are limited to numeric and string types.

If you have not programmed in one of those other languages, this will be your introduction to a powerful tool for formatting text.

A format expression is a string with embedded format specifiers. Anything that is not a format specifier is copied literally to the function output string. Each embedded format specifier is replaced with the value of an expression from the list, formatted according to the specifier. For example:

```
PRINT USING$("", "Pi is approximately %f.", PI())
```

 function call



Pi is approximately 3.141593.

printed output for English locale

The `<locale_exp>` is `""`, meaning "use my default locale".

The `<format_exp>`, "Pi is approximately %f", has one format specifier, "%f".

"%f" means, "use the default decimal floating point output format".

The expression list has one item, the math function **PI()**.

In the output, "%f" is replaced by the value of the **PI()** function.

Your output may be different if your locale language is not English.

## Format Specifiers

Here is a brief summary of the available format specifiers:

For this type of data	Use these formats	Comments
String	%s %S	%S forces output to upper-case
Number	%f %e %E %g %G %a %A	Standard BASIC! numbers are floating point Use %f for decimal output: "1234.567" Use %e or %E for exponential notation: "1.234e+03" %E writes upper-case: "1.234E+03" %g (%G) lets the system choose %f or %e (%E) %a and %A are "hexadecimal floating point"
Integer	%d %o %x %X	USING\$ can use some math functions as integers Use %d for decimal, %o for octal, %x %X for hex %x writes lower-case abcdef, %X writes upper-case
Special integer	%c %C %t	These specifiers can operate on an integer %c %C output a character, %C writes upper-case %t represents a family of time format specifiers
None	%% %n	These specifiers do not read the expression list %% writes a single "%" to the output %n writes a newline, exactly the same as \n

For more information about %a and %A, see the Android documentation linked above.

Android's %b and %B are not supported because BASIC! has no Boolean type.

Android's %h and %H hash code specifiers are limited to strings and numbers in BASIC!.

For an explanation of **USING\$()** with integer format specifiers, see below.

There is a whole family of time format specifiers: **%t<x>** where **<x>** is another letter. They operate on an integer, which they interpret as the number of milliseconds since the beginning of January 1, 1970, UTC (the "epoch"). You can apply time format specifiers to the output of the **TIME()** functions. Note, however, that the **%t** time specifiers use your local timezone, not the **TimeZone.set** value.

There are more than 30 time format specifiers. A few examples appear below, but to get the full list you should read the Android documentation linked above.

```

PRINT USING$("", "The time is: %tl:%<tM:%<tS %<Tp", time()) % the hard way
PRINT USING$("", "The time is: %tr", time()) % same thing!
02:27:16 PM example of printed output

t = TIME(2001, 2, 3, 14, 5, 6) % set 2001/02/03 14:05:06, local timezone
PRINT USING$("sv", "%tA", t) % day in Swedish, prints "lördag"
PRINT USING$("es", "%tB", t) % month in Spanish, prints "febrero"
PRINT USING$("", "%tY/%tm/%td", t, t, t) % prints "2001/02/03"
PRINT USING$( , "%tY/%<tm/%<td", t) % prints "2001/02/03"
PRINT USING$("", "%tF", t) % prints "2001:02:03"
PRINT USING$("en_GB", "%tH:%<tM:%<tS", t) % prints "14:05:06"
PRINT USING$("in_IN", "%tT", t) % prints "14:05:06"

```

Note: Date and time are printed for your local timezone, regardless of either the `TIMEZONE.SET` setting or the locale parameter. Try the same set of examples with **TIMEZONE.SET "UTC"**. Unless that is your local timezone, a different hour and perhaps even a different day will be displayed.

### Optional Modifiers

The format specifiers can be used exactly as shown in the table. They have default settings that control how they behave. You can control the settings yourself, fine-tuning the behavior to suit your needs.

You can modify the format specifiers with *index*, *flags*, *width*, and *precision*, as shown in this example:

%3\$-,15.4f						
"	3\$	-,	15	.	4	f
%						
	<index>	<flags>	<width>		<precision>	<specifier>

### Index

Normally the format specifiers are applied to the arguments in order. You can change the order with an argument index. An index a number followed by a \$ character. The argument index **3\$** specifies the third argument in the list.

```

PRINT USING$("", "%3$s %s %s", "a", "b", "c") % prints "c a b"

```

The special argument index "<" lets you reuse an argument.

```

PRINT USING$("", "%o %<d %<h", int(64) ) % prints "100 64 40"

```

In the last example, there is only one argument, but three format specifiers. This is not an error because the argument is reused.

### Flags

There are six flags:

-	left-justify; if no flag, right-justify
+	always show sign; if no flag, show "-" but do not show "+"
0	pad numbers with leading zeros; if no flag pad with spaces
,	use grouping separators for large numbers

(	put parentheses around negative values
#	alternate notation (leading 0 for octal, leading 0x for hexadecimal)

### Width

The **width** control sets the minimum number of characters to print. If the value to format is longer than the width setting, the entire value is printed (unless it would exceed the **precision** setting). If the value to format is shorter than the **width** setting, it is padded to fill the width. By default, it is padded with spaces on the left, but you change this with the "-" and "0" flags.

### Precision

The **precision** control means different things to different data types.

For floating point numbers, **precision** specifies the number of characters to print after the decimal point, padding with trailing zeros if needed.

For string values, it specifies the maximum number of characters to print. If **precision** is less than **width**, only **precision** characters are printed.

```
"%4s", "foo"    → " foo"
"%-4s", "foo"   → "foo "
"%4.2s", "foo"  → "fo"
```

The **precision** control is not valid for other types.

In the example above, **%,15.4f**:

The **flags** "-" and "," mean "left-justify the output" and "use a thousands separator".

The **width** is 15, meaning the output is to be at least 15 characters wide.

The **precision** is 4, so there will be exactly four digits after the decimal point.

The whole format specifier means, "format a floating point number (%f) left-justified (" -") in a space 15 characters wide, with 4 characters after the decimal point, with a thousands separator (",")."

The characters used for the decimal point and the thousands separator depend on the locale:

```
"1,234.5678"    " for locale "en"
"1 234,5678"    " for locale "fr"
"1.234,5678"    " for locale "it"
```

### Integer values

BASIC! has only double-precision floating point numbers. It does not have an integer type. The **USING\$()** function supports format specifiers ("%d", "%t", "%x", "%X") that apply only to integer values. *Is converting from type Double necessary, it will be done automatically.*

~~**USING\$()** has a special relationship with the math functions that intrinsically produce integer results. BASIC! converts the output of these functions to floating point, for storage in numeric variables, but **USING\$()** can get the original integer values. For example:~~

```
PRINT USING$("", "%d", 123)      % ERROR!
PRINT USING$("", "%d", INT(123)) % No error
```

The functions that can produce integer values for **USING\$()** are:

```
INT()  BIN()  OCT()  HEX()
CEIL() FLOOR()
ASCII() UCODE()
BAND() BOR() BXOR()
SHIFT() TIME()
```



## FORMAT\_USING\$(<locale\_sexp>, <format\_sexp> { , <exp>}...)

Alias for **USING\$()**. You can use the two equivalent functions to make your code easier to read.  
For example:

```
string$ = FORMAT_USING("", "pi is not %d", pi())  
Print USING$("en_US", "Balance: $%8.2f", balance)
```

## HEX\$(<nexp>|<color\_sexp>)

Returns a string representing the hexadecimal representation of the numeric expression.  
If the alternative <color\_sexp> is specified, the function returns the color by the (#) hex color notation like (#)ff00ff00 (\_Lime). The source specified by <color\_sexp> can be one of these options:

```
{Alpha,}Red,Green,Blue % All these members in a range from 0 to 255  
(comma delimited string)  
or  
_{Alpha,}ColorName  
({comma delim.} string)  
or  
#{hn}hnhnhn  
(hex. String)
```

Example:

```
hColor$ = "#" + HEX$("0,0,255") % (_Blue) returns "#ff0000ff"
```

## SPC\$(<nexp>{, <sexp>})

Returns a string with <nexp> spaces.

If <sexp> is set, the function returns <nexp> times the into <sexp> defined string.

For example, CHR\$(9) can insert tabs instead of spaces.

## ONEX\$(<locale\_sexp>,<nexp>)

Returns the Ordinal Number Extension of a given numeric value.

By default a point "." will be returned.

If <locale\_sexp> is "uk", "en" or "us" it returns for values > 0 "st", "nd", "rd" or "th".

"se" (Swedish) returns ":a", ":b" or ":e".

"ir" (Irish) returns "ú".

"nl" returns "e".

"lat" (Latin), "gl" (Galician), "it" (Italian), "pt" (Portuguese) or "sp" (Spanish) return only "o".

See also USING\$ for Locale expressions.

Examples:

ONEX\$("uk", 0) returns "."

ONEX\$("", 1) returns ".". If your default device language is maybe German

ONEX\$("se", 2) returns ":b"

ONEX\$("uk", 3) returns "rd"

ONEX\$("us", 11) returns "th"

ONEX\$("", 13) returns ".". If your default device language is maybe French

ONEX\$("uk", 112) returns "th"

ONEX\$("us", 1003) returns "nd"

## REVERSE\$(<sexp>)

Returns a copy of <sexp> with the order of the characters reversed.

Example:

PRINT Reverse\$("Was it a cat I saw?") % Returns "?was I tac a ti saW"

## REPLACE\$(<sexp>, <find\_sexp>, <replace\_sexp>{, <mode\_sexp>})

Returns <sexp> with all instances of <find\_sexp> replaced with <replace\_sexp>.

Options of <mode\_sexp>:

"\_Default"

replace **without** regular expressions how described above(default)

"\_RegEx\_All"

replace **with** regular expressions **all** instances of <argument\_sexp>

"\_RegEx\_First"

replace **with** regular expressions **first** instance of <argument\_sexp>

"\_RegEx\_All\_IgnoreCase"

replace **with** regular expressions **all** instances of <argument\_sexp>, but ignore case

"\_RegEx\_First\_IgnoreCase"

replace **with** regular expressions **first** instance of <argument\_sexp>, but ignore case

Examples:

```

Result$ = Replace$("abcdefghijklmnabc", "abc", "rst", "_RegEx_All")
!-- returns "rstdefghijklmnrst"
PRINT Result$
Result$ = Replace$("abc:defghi,jklmn;abc", "[,:;]", "|", "_RegEx_All")
!-- returns "abc|defghij|klmn|abc"
PRINT Result$
Result$ = Replace$("abcdefghijklmnabc", "abc", "rst", "_RegEx_First")
!-- returns "rstdefghijklmnabc"
PRINT Result$

```

## NTRIM\$(<nexp>)

Returns a number as a string, which is trimmed to the smallest possible string length.

See also INT\$()

**Join** <source\_array\$[]>, <result\_svar> {, <separator\_sexp>{, <wrapper\_sexp>}}

**Join.all** <source\_array\$[]>, <result\_svar> {, <separator\_sexp>{, <wrapper\_sexp>}}

**Join** <source\_array[]>, <result\_svar> {, <separator\_sexp>{, <wrapper\_sexp>}}

**Join.all** <source\_array[]>, <result\_svar> {, <separator\_sexp>{, <wrapper\_sexp>}}

The elements of the <source\_array\$[]> or <source\_array[]> are joined together as a single string in the <result\_svar>. [Using a numeric array all numbers will be trimmed to the smallest possible string length.](#) By default, the source elements are joined with nothing between them or around them.

You may specify optional modifiers that add characters to the string. Copies of the separator string <separator\_sexp> are written between source elements. Copies of the wrapper string <wrapper\_sexp> are placed before and after the rest of the result string.

The **Join** command omits any empty source elements. The **Join.all** command includes all source elements in the result string, even if they are empty. There is no difference between the two commands unless you specify a non-empty separator string. **Join.all** places copies of the separator between all of the elements, including the empty ones.

An example of an operation that uses both separators and wrappers is a CSV string, for "comma-separated values".

```

InnerPlanets$ = "Mercury Venus Earth Mars"
SPLIT IP$[], InnerPlanets$
JOIN IP$[], PlanetsCSV$, "\"\", \"\" \" % = CHR$(34) + "\", CHR$(34)
PRINT PlanetsCSV$

```

This prints the string **"Mercury","Venus","Earth","Mars"** (including all of the quotes). The separator puts the "," between planet names, and the wrapper puts the " at the beginning and end of the string.

**Split** <result\_array\$[]>, <sexp> {, <test\_sexp>}

**Split.all** <result\_array\$[]>, <sexp> {, <test\_sexp>}

Splits the source string <sexp> into multiple strings and place them into <result\_array\$[]>. The array is specified without an index. If the array exists, it is overwritten. Otherwise a new array is created. The result is always a one-dimensional array.

The string is split at each location where <test\_sexp> occurs. The <test\_sexp> occurrences are removed from the result strings. The <test\_sexp> parameter is optional; if it is not given, the string is split on whitespace. Omitting the parameter is equivalent to specifying "\s+". A whitespace is not only Space (ASCII 32). Is one of 25 Unicode characters. CR = carriage return LF = line feed are also whitespaces.

See also [https://en.wikipedia.org/wiki/Whitespace\\_character](https://en.wikipedia.org/wiki/Whitespace_character).

Thus WORD\$(), Split and Split.all have to test against 25 characters instead of one.

That slows down the speed of this command. Use " " as <test\_sexp> if you want to split by a simple Space character.

If the beginning of the source string matches the test string, the first element of the result array will be an empty string. This differs from the **WORD\$()** function, which strips leading and trailing occurrences of the test string from the source string before splitting.

Two adjacent occurrences of the test expression in the source expression result in an empty element somewhere in the result array. The **Split** command discards these empty strings if they occur at the end of the result array. To keep these trailing empty strings, use the **Split.all** command.

Example:

```
string$ = "a:b:c:d"
delimiter$ = ":"
SPLIT result[], string$, delimiter$
```

```
ARRAY.LENGTH length, result[]
FOR i = 1 TO length
PRINT result[i] + " ";
NEXT i
PRINT ""
```

Prints: a b c d

Note: The <test\_sexp> is actually a Regular Expression. If you are not getting the results that you expect from the <test\_sexp> then you should examine the rules for Regular Expressions at:

<http://developer.android.com/reference/java/util/regex/Pattern.html>

For the Regular Expression "[afl]" you need a (Java) string "\\[afl\\]"

A good address to verify is:

<http://www.regexplanet.com/advanced/java/index.html>

If you miss a **TALLY** command, do so:

```
FN.DEF TALLY( mString$, regExp$)
  SPLIT.ALL result_array[], mString$, regExp$
  ARRAY.LENGTH count, result_array[]
  ARRAY.DELETE result_array$[]
  FN.RTN count - 1
FN.END
```



```

Count = TALLY("abcdefghijklmnabc", "abc") %-- returns 2
PRINT Count
Count = TALLY("abcdefghijklmnabc", "\\[afl\\]") %-- returns 0
PRINT Count
Count = TALLY("abcdefghijklmnabc", "[afl]") %-- returns 4
PRINT Count
Count = TALLY("abc Defghi Jklmn Abc", "[:upper:]") %-- returns 3
PRINT Count
Count = TALLY("abc Defghi Jklmn Abc", "[:lower:]") %-- returns 14
PRINT Count
Count = TALLY("abc Defghi Jklmn Abc", "[:space:]") %-- returns 3
PRINT Count

```

### Array.sort Array[{<start>,<length>}]

Sorts the values of the specified array (Array[] or Array\$[]) or array segment (Array[start,length] or Array\$[start,length]) in ascending order.

### Array.search Array[{<start>,<length>}], <value\_exp>, <result\_nvar>{,<start\_nexp>}

### Array.search Array\$[{<start>,<length>}], <value\_exp>, <result\_nvar>{,<start\_nexp>}

Searches in the numeric or string array (Array[] or Array\$[]) or array segment (Array[start,length] or Array\$[start,length]) for the specified numeric or string value, which may be an expression. If the value is found in the array, its position will be returned in the result numeric variable <result\_nvar>. If the value is not found the result will be zero.

If only a segment of an array is used, the result is in conjunction to the start point.

Example:

```

Array.load ar[],10, 8,12, 20, 6, 7, 88, 11
Array.search ar[4, 3], 6, pos % pos returns 2 instead of 5
Array.search ar[4, 3], 88, pos % pos returns 0, because 88 is not part of the segment

```

### Array.binary.search Array[]|Array\$[], <value\_exp>, <result\_nvar>

Searches in the numeric or string array (Array[] or Array\$[]) for the specified numeric or string value, which may be an expression. If the value is found in the array, its position will be returned in the result numeric variable <result\_nvar>. If the value is not found the result will be zero.

This command use the very fast binary-search-method. **Keep in mind, that you have to sort the array before to prevent unexpected results.**

Example:

```

Array.sort myTownArray$[]
Array.binary.search myTownArray$[], "Paris", resultIndex

```

### Array.from.string <ssexp>, Array[]

Converts Characters of Strings into an Array of numbers like the UCODE() function.

Example:

```
Array.from.string s$, nA[]
```

### **Array.to.string <sexp>, Array[]**

Converts the numbers of a numeric array into a String like the CHR\$() function.

Example:

```
Array.to.string s$, nA[]
```

**Array.Rnd Array[] {{{{{{, <length\_nexp>}, <low\_nexp>}, <high\_nexp>},  
<seed\_nexp>}, <type\_nexp>}, <generator\_nexp>}}**

Creates a numeric array with pseudo-random numbers by using a new random generator. The array length is described by the optional <length\_nexp>. If not given an array with only one member is returned.

The optional <low\_nexp> sets the starting value or the offset for Gaussian random numbers. Default is 0.

The optional <high\_nexp> sets the ending value or the scale for Gaussian random numbers. Default is 1.

If <low\_nexp> = 0 <high\_nexp> is the scale factor in conjunction to RND().

The optional <seed\_nexp> sets the seed. Default is 0 for unpredictable and not reproducible numbers. Seed number > 0 create reproducible results.

The optional <type\_nexp> sets the returned type of the random numbers. Default is 0 for normal random numbers. If <type\_nexp> is 1, Gaussian random numbers are returned. If <type\_nexp> is 2, boolean random numbers are returned.

The optional <generator\_nexp> sets the random generator. Default is 0 for the standard generator. If <generator\_nexp> is 1, a secure random number generator (RNG) implementing the default random number algorithm will be constructed. Only usable for Android 4.4+ (KitKat+).

Example 1:

```
Array.Rnd r[], 2, 0, 100, 4711, 0
Print r[1] % Returns 4.315872869138159
Print r[2] % Returns 37.83154009540295
Array.Rnd nr[]
Print nr[1] % Returns an unpredictable and not reproducible number between 0 and < 1.
```

Example 2:

```
offs=20 : fac=10
Array.Rnd rnd[],1, offs, offs + fac
! The results will be between 20 and 30
Print rnd[1], offs + RND() * fac
Array.Rnd rnd[],1, offs, offs + fac, 0, 2 % Boolean result
! Now the result will be 20 or 30
Print rnd[1]
```

**Array.by.index SourceArray[], IndexArray[], DestinationArray[]**

**Array.by.index SourceArray\$[], IndexArray[], DestinationArray\$[]**

Copies elements of an existing SourceArray[] to the DestinationArray[], which are specified by the IndexArray[].

Example:

```
ARRAY.LOAD values[], 1, 2.45, 3, 4, 5, 6
ARRAY.LOAD idx[], 1, 4, 2, 3, 1, 5, 2, 6, 2
ARRAY.BY.INDEX values[], idx[], out[]
JOIN.ALL out[], result$, " , "
PRINT result$ % Returns "1, 4, 2.45, 3, 1, 5, 2.45, 6, 2.45"
```

**Array.truth.choice SourceTrueArray[], SourceFalseArray[], IndexArray[], DestinationArray[]**

**Array.truth.choice SourceTrueArray\$[], SourceFalseArray\$[], IndexArray[], DestinationArray\$[]**

Copies elements of existing SourceTrueArray[] or SourceFalseArray[] to the DestinationArray[], depended on the specifications of the IndexArray[]. If the corresponding item of the IndexArray[] is 0 the item of the SourceFalseArray\$[] will be copied. Otherwise in case of a item <> 0 in the IndexArray[] the SourceTrueArray\$[] will be copied. IndexArray[], SourceTrueArray[] and SourceFalseArray[] have to be with the same count of items.

Example:

```
ARRAY.LOAD valuesTrue[], 1, 2.45, 3, 4, 5, 6
ARRAY.LOAD valuesFalse[], 6, 5, 4, 3, 2.45, 1
ARRAY.LOAD idx[], 1, 0, 0, 0, 0, 1
ARRAY.TRUTH.CHOICE valuesTrue[], valuesFalse[], idx[], out[]
JOIN.ALL out[], result$, " , "
PRINT result$ % Returns "1, 5, 4, 3, 2.45, 6"
```

**Array.truth.subset <ls\_truth\_nexp>, SourceArray[], IndexArray[], DestinationArray[]**

**Array.truth.subset <ls\_truth\_nexp>, SourceArray\$[], IndexArray[], DestinationArray\$[]**

Copies elements of an existing SourceArray[] to the DestinationArray[], which are specified by the IndexArray[]. If <ls\_truth\_nexp> is 0, all source entire in conjunction to IndexArray[] members which are 0 are copied. If <ls\_truth\_nexp> is **not** 0, all source elements in conjunction to IndexArray[] members which are **not** 0 are copied.

Example:

```
ARRAY.LOAD in1[], 1,2,27,4,5,6,7,8,3,4,57,114,115
ARRAY.LOAD in2[], 0,0,1 ,0,0,1,0,0,1,0,1 ,1 ,0
ARRAY.TRUTH.SUBSET 1, in1[],in2[],out[]
! -> out[] will be: 27, 6, 3, 57, 114
ARRAY.TRUTH.SUBSET 0, in1[],in2[],out[]
! -> out[] will be: 1, 2, 4, 5, 7, 8, 4, 115
ARRAY.LOAD in3$[], "1", "2", "27", "4", "5", "6", "7", "8", "3", "4", "57", "114", "115"
ARRAY.TRUTH.SUBSET 0, in3$[],in2[],out2$[]
! -> out2$[] will be: 1, 2, 4, 5, 7, 8, 4, 115
```

### Array.truth.index <ls\_truth\_nexp>, IndexArray[], DestinationArray[]

Copies elements of an existing IndexArray[] to the DestinationArray[], which are specified by the IndexArray[]. If <ls\_truth\_nexp> is 0, all index entries in conjunction to IndexArray[] members which are 0 are copied. If <ls\_truth\_nexp> is **not** 0, all source elements in conjunction to IndexArray[] members which are **not** 0 are copied.

Example:

```
ARRAY.LOAD in2[], 0,0,1,0,0,1,0,0,1,0,1,1,0
ARRAY.TRUTH.INDEX 1, in2[], out[]
! -> out[] will be: 3, 6, 9, 11, 12
ARRAY.TRUTH.INDEX 0, in2[], out[]
! -> out[] will be: 1, 2, 4, 5, 7, 8, 10, 13
```

### Array.to.dims SourceArray[], DimensionArray[], DestinationArray[]

#### Array.to.dims SourceArray\$[], DimensionArray[], DestinationArray\$[]

Copies all elements of an existing SourceArray[] to the DestinationArray[], which is specified in order by the DimensionArray[]. The number of source and destination array elements must be the equal. If the Destination Array exists, it will be overwritten. If the Destination Array does not exist, a new array is created. The arrays may be either numeric or string arrays but they must both be of the same type.

Example:

```
Array.load s[], 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 % 10 elements
Array.load d[], 5, 2 % 5 * 2 = 10
Array.to.dims s[], d[], r[]
Print r[1, 1] % Returns 0.0
Print r[5, 2] % Returns 9.0
```

\	1	2	3	4	5
1	0	2	4	6	8
2	1	3	5	7	9

### Array.row.print SourceArray[], {{<lineNum\_nexp>}, <result\_svar>}

#### Array.row.print SourceArray\$[], {{<lineNum\_nexp>}, <result\_svar>}

Prints an Array as rows into the console or String. The order is defined by the given dimensions of the Array. If <lineNum\_nexp> is greater than 0, line numbers are added at the start of the row. Default is 1. Is <result\_svar> given optionally, the rows are transferred into a String with line feed endings ("\n"). In this case no console output is returned.

Example:

```
Array.load xyzPoints[], 0,0,0, 250,100,0, 500,0,0, 500,500,0, 250,400,0, 0,500,0
! Returns rows with line numbers and x, y, z values → 1: 0.0, 0.0, 0.0 ...
Array.row.print xyzPoints[], 1
```

## Array.Mat Command Group

Mat stands for a Matrix, a two-dimensional array.

Matrices, the majority of matrix are usually shown with capital letters such as **A** or **B**. Their number of columns with **n** and the number of rows with **m**.



An  $m \times n$  matrix: the  $m$  rows are horizontal and the  $n$  columns are vertical. Each element of a matrix is often denoted by a variable with two subscripts. For example,  $a_{2,1}$  represents the element at the second row and first column of the matrix.

Source: Wikipedia

Example:

```
Array.load s[], 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 % 10 elements
Array.load d[], 5, 2 % 5 * 2 = 10
Array.to.dims s[], d[], r[]
```

$r[] \rightarrow [5,2]$  is a matrix defined by columns (5) and rows (2), **but the dimension positions of rows "y" and columns "x" are reversed contrary to the mathematical notation rows "n" and columns "m"**. The order in BASIC! is maybe different to other programming languages but the same as in FORTRAN. The advantage is the result, it corresponds in example to the order ( [ width (x), height (y) ] ) of the bitmap or screen pixels.

→ x  
↓  
y

\	1	2	3	4	5
1	0	2	4	6	8
2	1	3	5	7	9

If you need an order like the example below use `_Toggle` (Skill of `Array.Mat.Skill`) or `Array.Mat.Toggle`.

\	1	2	3	4	5
1	0	1	2	3	4
2	5	6	7	8	9

`Array.copy r [1,4], c[]`

It equals to `Array.copy s [1,4], c[]` because it is flattened to a vector and has lost its dimensions except for one.

$c \rightarrow [4]$  it is a Vector, it has only one dimension, it has by definition no columns or rows.

\	1	2	3	4	
1	0	1	2	3	?
2	1				
3	2				
4	3				
	?				

If the input needs a Matrix, the Vector elements are placed as a **column** by default.

$c \rightarrow C$

\	1
1	0
2	1
3	2
4	3

If you need a Matrix **row** input use `_Transpose` (Skill of `Array.Mat.Skill`) or `Array.Mat.Transpose`.

\	1	2	3	4
1	0	1	2	3

A Scalar is a special case of a Vector with only one element. In a Matrix it has only one column and one row.

$C[4] \rightarrow$

\	1
1	3

Single numeric values are automatically converted into a Scalar, thus it is processed as an array with only one element.



## Array.Mat.Toggle SourceArray[], {<direction\_sexp>}, DestinationArray[]

### Array.Mat.Toggle SourceArray\$[], {<direction\_sexp>}, DestinationArray\$[]

Mat stands for a Matrix, a two-dimensional array.

Copies all elements of an existing SourceArray[] to the DestinationArray[], but toggles the column and row order over the element [1,1].

The direction is specified by the optional <direction\_sexp>.

- With the character string "\_CtoR" a **column** packed array can be copied into a **row** packed array. This is the default direction.
- With the character string "\_RtoC" a **row** packed array can be copied into a **column** packed array.

The sum of the source and target array elements is the same. If the target array is present, it will be overwritten. If the target array does not exist, a new array is created. The arrays can be either numeric or string arrays, but both must be of the same type.

In the case of square matrices, the results of Array.Mat.Toggle and Array.Mat.Transpose are the same.

Example:

```
ARRAY.LOAD s[], ~
0, 1, 2, 3, 4, ~
5, 6, 7, 8, 9 % 10 elements
Array.load d[], 5, 2 % 5 * 2 = 10
Array.to.dims s[], d[], r[]
Array.Mat.Toggle r[], "_CtoR", n[]
Array.dims n[], d[] % d[] returns 5, 2 as usual
```

Before toggling:

\	1	2	3	4	5
1	0	2	4	6	8
2	1	3	5	7	9

After toggling:

\	1	2	3	4	5
1	0	1	2	3	4
2	5	6	7	8	9

## Array.Mat.Transpose SourceArray[], DestinationArray[]

### Array.Mat.Transpose SourceArray\$[], DestinationArray\$[]

Mat stands for a Matrix, a two-dimensional array.

Copies all elements of an existing SourceArray[] to the DestinationArray[], but transposes over the element [1,1]. The sum of the source and target array elements is the same. If the target array is present, it will be overwritten. If the target array does not exist, a new array is created. The arrays can be either numeric or string arrays, but both must be of the same type.

In the case of square matrices, the results of Array.Mat.Toggle and Array.Mat.Transpose are the same.

Example:

```
Array.load s[], 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 % 10 elements  
Array.load d[], 5, 2 % 5 (Columns) * 2 (Rows) = 10  
Array.to.dims s[], d[], r[]
```

**! How to store the third column of Array r[] (Matrix) into a list**

```
columnInArrayR = 3 : nRowsInArrayR = 2 % In r[] !  
List.create n, vertical  
List.add.array vertical, r[1 + (columnInArrayR - 1) * nRowsInArrayR, nRowsInArrayR]
```

**! How to store the second row of Array r[] (Matrix) into a list**

```
Array.Mat.Transpose r[], n[]  
Array.dims n[], d[] % d[] returns 2, 5 instead 5, 2  
  
rowInArrayR = 2 : nColumnsInArrayR = 5 % In r[] !  
List.create n, horizontal  
List.add.array horizontal, n[1 + (rowInArrayR - 1) * nColumnsInArrayR, nColumnsInArrayR]
```

Before transposing:

\	1	2	3	4	5
1	0	2	4	6	8
2	1	3	5	7	9

After transposing:

\	1	2
1	0	1
2	2	3
3	4	5
4	6	7
5	8	9

## **Array.Mat.Skill <Bundle\_Pointer\_nexp>{, <runtime\_error\_nexp>}**

Mat stands for a Matrix, a two-dimensional array.

Arguments and results are stored in a Bundle specified by <Bundle\_Pointer\_nexp>.

The Bundle key "\_Result" stores the result. The Bundle key "\_Error" logs errors if any occurs.

If <runtime\_error\_nexp> is specified by a value > 0, the execution is aborted immediately in a case of an error. The default is 1. If the value is 0 the execution will not stopped if possible.

### **LEVEL 1**

A skill is specified by a Bundle entry like this

```
BUNDLE.PUT sBptr, "_Skill", "_min('ArrayLeft', 'ArrayRight')"
```

The operator begins with an underscore and its arguments within parentheses.

The key names of the arguments have to be within **single quotes**.

The following four characters ) : ( , ' cannot be between single quotes.

```
BUNDLE.PUT sBptr, "ArrayLeft", mArrayLeft[]
```

```
BUNDLE.PUT sBptr, "ArrayRight", mArrayRight[]
```

After successful execution a result is returned by the Bundle key "\_Result".

```
BUNDLE.GET sBptr, "_Result", mResult[]
```

Otherwise this key is deleted or not be created. In this case the Bundle key "\_Error" returns the error log.

The result of an operation can be a numeric Vector (one-dimensional Array) or a Matrix (two-dimensional Array). If a single value is the default result, a one-dimensional array with length 1 is returned.

All possibilities of **Array.math** are supported, thus operations on Level 1 return only Arrays.

<b>LEVEL 1 skills</b>		
<b>_Skill Syntax</b>	<b>Result</b>	<b>Description</b>
	<b>_Return</b>	<b>Holds the Result as an Array or Bundle</b>
All <b>Array.math</b> operations like <b>_min('left', 'right')</b>	Array specified by the dimensions of the first argument. (Scalar, Vector, Matrix)	<b>Operates member by member</b>

### **LEVEL 2**

The result of an operation before can be used as an argument within the next one.

```
BUNDLE.PUT sBptr, "_Skill", "_*('_Result', '_Result')"
```

```
BUNDLE.GET sBptr, "_Result", mResult[] % Returns the square of each entry.
```

```
BUNDLE.PUT sBptr, "_Skill", "_Copy('ArrayLeft', 'Mem1')"
```

 % Copies numeric content.

LEVEL 2 skills		
_Skill Syntax	Result	Description
_Copy('A', 'target')		<b>Copies numeric content.</b> Copy numeric content from a Bundle location specified by a Bundle key into a different Bundle location. This operation works only inside the Bundle which is given by the command call. Copying bundles is not intended!

### LEVEL 3

The JAMA : The Java Matrix Package (<https://math.nist.gov/javanumerics/jama/>) is used in the background if needed.

At Level 3 results from type Bundle are returned also.

To get access at these values proceed as follows

BUNDLE.GB sBptr, "\_Result", resPtr % Returns the main Bundle pointer

BUNDLE.GET resPtr, "\_EigenvalueMatrix", EigenvalueMatrix[] % Returns the Matrix

or

! Copy numeric content from a Bundle into a different main Bundle location.

! Copying bundles is not intended!

BUNDLE.PUT sBptr, "\_Skill", "\_Copy('\_Result': '\_EigenvalueMatrix', 'Mem1')"

LEVEL 3 skills																																						
_Skill Syntax	Result	Description																																				
Elementary Operations																																						
_Normalize('A')	Array (Matrix)	<b>Normalizes</b> a matrix to make the elements sum to 1.																																				
_ScalarMulti('A', 's')	Array (Matrix)	<b>Scalar Multiplication</b> Multiply a Matrix by a Scalar, $A * s$																																				
_Times('A', 'B')	Array (Matrix)	<b>Matrix Multiplication</b> Linear algebraic matrix multiplication Returns Matrix product, $A * B$																																				
_*('a', 'b')	Array (Matrix or Vector)	<b>Dot Product</b> See level 1																																				
_x3d('a', 'b')	Array (Matrix or Vector)	<b>Cross Product</b> See level 1																																				
_ToggleCtoR('A')	Array (Matrix)	<b>ToggleCtoR</b> Before toggling: <table><tr><td>\</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr><tr><td>1</td><td>0</td><td>2</td><td>4</td><td>6</td><td>8</td></tr><tr><td>2</td><td>1</td><td>3</td><td>5</td><td>7</td><td>9</td></tr></table> After toggling: <table><tr><td>\</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr><tr><td>1</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td></tr><tr><td>2</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td></tr></table>	\	1	2	3	4	5	1	0	2	4	6	8	2	1	3	5	7	9	\	1	2	3	4	5	1	0	1	2	3	4	2	5	6	7	8	9
\	1	2	3	4	5																																	
1	0	2	4	6	8																																	
2	1	3	5	7	9																																	
\	1	2	3	4	5																																	
1	0	1	2	3	4																																	
2	5	6	7	8	9																																	
_ToggleRtoC('A')	Array (Matrix)	<b>ToggleRtoC</b> Before toggling: <table><tr><td>\</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr><tr><td>1</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td></tr><tr><td>2</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td></tr></table> After toggling: <table><tr><td>\</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr><tr><td>1</td><td>0</td><td>2</td><td>4</td><td>6</td><td>8</td></tr><tr><td>2</td><td>1</td><td>3</td><td>5</td><td>7</td><td>9</td></tr></table>	\	1	2	3	4	5	1	0	1	2	3	4	2	5	6	7	8	9	\	1	2	3	4	5	1	0	2	4	6	8	2	1	3	5	7	9
\	1	2	3	4	5																																	
1	0	1	2	3	4																																	
2	5	6	7	8	9																																	
\	1	2	3	4	5																																	
1	0	2	4	6	8																																	
2	1	3	5	7	9																																	

LEVEL 3 skills																																						
_Skill Syntax	Result	Description																																				
<code>_Transpose('A')</code>	Array (Matrix)	<b>Transpose</b> Before transposing: <table><tr><td>\</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr><tr><td>1</td><td>0</td><td>2</td><td>4</td><td>6</td><td>8</td></tr><tr><td>2</td><td>1</td><td>3</td><td>5</td><td>7</td><td>9</td></tr></table> After transposing: <table><tr><td>\</td><td>1</td><td>2</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>2</td><td>2</td><td>3</td></tr><tr><td>3</td><td>4</td><td>5</td></tr><tr><td>4</td><td>6</td><td>7</td></tr><tr><td>5</td><td>8</td><td>9</td></tr></table>	\	1	2	3	4	5	1	0	2	4	6	8	2	1	3	5	7	9	\	1	2	1	0	1	2	2	3	3	4	5	4	6	7	5	8	9
\	1	2	3	4	5																																	
1	0	2	4	6	8																																	
2	1	3	5	7	9																																	
\	1	2																																				
1	0	1																																				
2	2	3																																				
3	4	5																																				
4	6	7																																				
5	8	9																																				
<code>_UnaryMinus('A')</code>	Array (Matrix)	<b>Unary minus</b> Returns -A																																				
Decompositions																																						
<code>_Cholesky('A' {'B'})</code>	Bundle [ { "_L", Array (Matrix) }, { "_X", Array (Matrix, optional) }, { "_IsSpd", Array[1] (Scalar) } ]	<b>Cholesky Decomposition of symmetric, positive definite matrices</b>  For a symmetric, positive definite matrix A, the Cholesky decomposition is a lower triangular matrix L so that $A = L * L'$ .  B a Matrix with as many rows as A and any number of columns. Returns X so that $L * L' * X = B$  If the matrix is not symmetric or positive definite, the constructor returns a partial decomposition and sets an internal flag that may be queried by the <code>_IsSpd</code> method. Returns a scalar array of one member. If 1 it is true. If 0 it is false.																																				

LEVEL 3 skills		
_Skill Syntax	Result	Description
<code>_Lu('A' {'B'})</code>	<pre>Bundle [   { "_L", Array (Matrix) },   { "_U", Array (Matrix) },   { "_Pivot", Array (Vector) },   { "_X", Array (Matrix, optional) },   { "_Determinant", Array[1] (Scalar) },   { "_IsNonsingular", Array[1] (Scalar) } ]</pre>	<p><b>LU Decomposition (Gaussian elimination) of rectangular matrices</b></p> <p>For an m-by-n matrix A with <math>m \geq n</math>, the LU decomposition is an m-by-n unit lower triangular matrix L, an n-by-n upper triangular matrix U, and a permutation vector piv of length m so that <math>A(\text{piv},:) = L*U</math>. If <math>m &lt; n</math>, then L is m-by-m and U is m-by-n.</p> <p>The LU decomposition with pivoting always exists, even if the matrix is singular, so the constructor will never fail. The primary use of the LU decomposition is in the solution of square systems of simultaneous linear equations. This will fail if <code>isNonsingular()</code> returns 0.0 (false).</p>
<code>_Qr('A' {'B'})</code>	<pre>Bundle [   { "_H", Array (Matrix) },   { "_Q", Array (Matrix) },   { "_R", Array (Matrix) },   { "_X", Array (Matrix, optional) },   { "_IsFullRank", Array[1] (Scalar) } ]</pre>	<p><b>QR Decomposition of rectangular matrices</b></p> <p>For an m-by-n matrix A with <math>m \geq n</math>, the QR decomposition is an m-by-n orthogonal matrix Q and an n-by-n upper triangular matrix R so that <math>A = Q*R</math>.</p> <p>The QR decomposition always exists, even if the matrix does not have full rank, so the constructor will never fail. The primary use of the QR decomposition is in the least squares solution of nonsquare systems of simultaneous linear equations. This will fail if <code>isFullRank()</code> returns 0.0 (false).</p>

LEVEL 3 skills		
_Skill Syntax	Result	Description
<code>_Svd('A')</code>	<pre>Bundle [ { "_S", Array (Matrix) }, { "_U", Array (Matrix) }, { "_V", Array (Matrix) }, { "_SingularValues", Array (Vector) }, { "_SingularValues", Array (Vector) }, { "_Norm2", Array[1] (Scalar) }, { "_Condition", Array[1] (Scalar) }, { "_Rank", Array[1] (Scalar) } ]</pre>	<p><b>Singular Value Decomposition of rectangular matrices</b></p> <p>For an m-by-n matrix A with <math>m \geq n</math>, the singular value decomposition is an m-by-n orthogonal matrix U, an n-by-n diagonal matrix S, and an n-by-n orthogonal matrix V so that <math>A = U \cdot S \cdot V'</math>.</p> <p>The singular values, <math>\sigma[k] = S[k][k]</math>, are ordered so that <math>\sigma[0] \geq \sigma[1] \geq \dots \geq \sigma[n-1]</math>.</p> <p>The singular value decomposition always exists, so the constructor will never fail. The matrix condition number and the effective numerical rank can be computed from this decomposition.</p>



LEVEL 3 skills		
_Skill Syntax	Result	Description
_Eigen('A')	<pre>Bundle [ { "_BlockDiagonalEigenvalue Matrix", Array (Matrix) }, { "_EigenvalueMatrix", Array (Matrix) }, { "_ImaginaryPartsEigenvalu es", Array (Vector) }, { "_RealPartsEigenvalues", Array (Vector) } ]</pre>	<p><b>Eigenvalue Decomposition of symmetric and square matrices</b></p> <p>If A is symmetric, then <math>A = V \cdot D \cdot V'</math> where the eigenvalue matrix D is diagonal and the eigenvector matrix V is orthogonal. I.e. <math>A = V \cdot \text{times}(D \cdot \text{times}(V \cdot \text{transpose}()))</math> and <math>V \cdot \text{times}(V \cdot \text{transpose}())</math> equals the identity matrix.</p> <p>If A is not symmetric, then the eigenvalue matrix D is block diagonal with the real eigenvalues in 1-by-1 blocks and any complex eigenvalues, <math>\lambda + i \cdot \mu</math>, in 2-by-2 blocks, <math>[\lambda, \mu; -\mu, \lambda]</math>. The columns of V represent the eigenvectors in the sense that <math>A \cdot V = V \cdot D</math>, i.e. <math>A \cdot \text{times}(V)</math> equals <math>V \cdot \text{times}(D)</math>. The matrix V may be badly conditioned, or even singular, so the validity of the equation <math>A = V \cdot D \cdot \text{inverse}(V)</math> depends upon <math>V \cdot \text{cond}()</math>.</p>
Equation Solutions		



LEVEL 3 skills		
_Skill Syntax	Result	Description
<code>_Rank('A')</code>	Array[1] (Scalar)	<b>Rank of a Matrix</b>
<code>_IsSymmetric('A')</code>	Array[1] (Scalar)	<b>Is Symmetric</b> Returns 0.0 for false and 1.0 for true.
<code>_Trace('A')</code>	Array[1] (Scalar)	<b>Sum of the diagonal elements</b>
<code>_Get('A', 'row', 'column')</code> (Math notation/order! )	Array[1] (Scalar)	<b>Get the value</b> at m = Number of row. n = Number of column.?
<code>_SubMatrix('A', 'rows', 'columns')</code> (Math notation/order! )	Array (Matrix)	<b>SubMatrix</b> Returns a Sub Matrix specified by rows = {start, end} (Array), columns = {start, end} (Array)?
<code>_RowPacked</code>	Array (Vector)	<b>RowPacked</b> Returns a Row Packed Array of the Matrix specified
<code>_ColumnPacked</code>	Array(Vector)	<b>ColumnPacked</b>
<code>_Rows('A')</code>	Array[1] (Scalar)	<b>Rows</b> Returns the number of rows.
<code>_Columns('A')</code>	Array[1] (Scalar)	<b>Columns</b> Returns the number of columns.

## LEVEL 4

A bundle entry "\_SkillArray" deals with a number of skills.

The pros are:

- You can execute Skill by Skill like `_sin('m1', 'm2')`, `_exp(...)`....
- Bypassing the interpreter (Speed)
- Programmable

The cons are:

- You should have a good plan
- Should be tested before in small parts
- Exception handling is more difficult

If a Bundle entry "\_Skill" is also in the Bundle its content will be overwritten by the current operated Array Skill.

If a Bundle entry "\_Counter" is in the main Bundle, it returns the current used Array Skill index.

Example:

```
FN.DEF outputArray(n[])
  ARRAY.DIMS n[], d[]
  ARRAY.LENGTH a[], d[]
  IF a[] = 2 % Only Matrices
    FOR r = 1 TO d[2]
      FOR c = 1 TO d[1]
        PRINT round (n[c, r], 4), ""
      NEXT
      PRINT " row", r
    NEXT
  ELSE
    FOR r = 1 TO d[1]
      PRINT ROUND(n[r], 4), ""
      PRINT "row: " + INT$(r)
    NEXT
  ENDIF
FN.END

FN.DEF getResult(sBptr)
  BUNDLE.GET sBptr, "_Counter", counter
  PRINT "_Counter", counter
  BUNDLE.GET sBptr, "_Skill", Skill$
  PRINT "_Skill", Skill$
  BUNDLE.CONTAIN sBptr, "_Error", isError
  IF isError
    BUNDLE.GET sBptr, "_Error", mError$
    PRINT "_Error !", mError$
  ENDIF
  BUNDLE.CONTAIN sBptr, "_Status", isStatus
  IF isStatus
    BUNDLE.GET sBptr, "_Status", mError$
    PRINT "_Status", mError$
  ENDIF
  BUNDLE.CONTAIN sBptr, "_Result", isResult
  IF isResult
    BUNDLE.TYPE sBptr, "_Result", keyType$
```

```

PRINT "_Result", keyType$
IF IS_IN("N", keyType$)
  BUNDLE.GET sBptr, "_Result", copyRes[]
  outputArray(copyRes[])
ELSE
  BUNDLE.GB sBptr, "_Result", rBptr
  DEBUG.ON
  DEBUG.DUMP.BUNDLE rBptr
  DEBUG.OFF
ENDIF
ENDIF
FN.END

```

#### Example How To:

```

Array.load sT2[], ~
0, 1, 2, ~
3, 4, 5, ~
6, 7, 8 % 9 elements
REDIM 1, sT2[3,3]
outputArray(copyRes[])
ARRAY.MAT.TOGGLE sT2[], "_CtoR", sT2[]
outputArray(copyRes[])
BUNDLE.PUT sBptr, "A", sT2[]
! ARRAY.LOAD mSkillArray$, "_Copy('A', 'B')", "_Copy('A', 'F')
ARRAY.LOAD mSkillArray$, "_Transpose('A')", "_Transpose('_Result')
BUNDLE.PUT sBptr, "_SkillArray", mSkillArray$[]
ARRAY.MAT.SKILL sBptr, 1
getResult(sBptr)

```

#### Example Multiple Linear Regression:

```

Array.Load xKgr[], 156.3, 158.9, 160.8, 179.6, 156.6, 165.1, 165.9, 156.7, 167.8, ~
160.8 % KOERPERGROESSE / body height in cm
Array.Load xKge[], 62, 52, 83, 69, 74, 52, 77, 65, 79, ~
51 % KOERPERGEWICHT / body weight in kg
Array.Load xAlt[], 24, 34, 26, 51, 43, 33, 22, 21, 19, 34 % ALTER / age in years
Array.Load y[], 47.1, 46.8, 49.3, 53.2, 47.7, 49.0, 50.6, 47.1, 51.7, ~
47.8 % RINGGROESSE / ring size in mm
Array.length aLY, y[]
DIM ones[aLY]
Array.Fill ones[], 1
List.create n, mm
List.add.Array mm, ones[]
List.add.Array mm, xKgr[]
! List.add.Array mm, xKge[] % Comment it out to get a multiple result
! List.add.Array mm, xAlt[] % Comment it out to get a multiple result
List.toArray mm, x[]
Array.length aLX, x[]
Array.length aLY, y[]
REDIM 1, x[aLX/aLY,aLY]
outputArray(x[])

```

```

BUNDLE.PUT sBptr, "X", X[]
BUNDLE.PUT sBptr, "Y", Y[]
PRINT " Conventional regularization  $b=(X^T * X)^{-1} * X^T * y$ :"
ARRAY.LOAD mSkillArray$, ~
" _Transpose('X')", ~
" _Copy(' _Result', 'Xt')", ~
" _Times('Xt', 'X')", ~
" _Inverse(' _Result')", ~
" _Times(' _Result', 'Xt')", ~
" _Times(' _Result', 'Y')"
```

BUNDLE.PUT sBptr, "\_SkillArray", mSkillArray\$[]  
 ARRAY.MAT.SKILL sBptr, 1  
 getResult(sBptr)

```

PRINT " Tikhonov regularization :"
```

BUNDLE.PUT sBptr, "lambda", 0 % If lambda 0 the same result as the conventional  
 ARRAY.LOAD mSkillArray\$, ~  
 ! " \_ToggleCtoR('X')", ~ % If needed  
 " \_Regress('X', 'Y', 'lambda')"

BUNDLE.PUT sBptr, "\_SkillArray", mSkillArray\$[]  
 ARRAY.MAT.SKILL sBptr, 1  
 getResult(sBptr)

```

BUNDLE.GET sBptr, " _Result", result[]
a = result[1]
b = result[2]
```

```

Array.min minX, xKgr[]
Array.max maxX, xKgr[]
leftY = a + b * minX
rightY = a + b * maxX
PRINT "y = a + b * x"
PRINT "y = " + Str$(Round(a, 4)) + " + " + Str$(Round(b, 4)) + " * x"
PRINT "y (Xmin) = " + Str$(Round(leftY, 4))
PRINT "y (160) = " + Str$(Round(a + b * 160, 4))
PRINT "y (170) = " + Str$(Round(a + b * 170, 4))
```

Source:  
 Taken from <https://www.crashkurs-statistik.de/einfache-lineare-regression/#berechnen>  
 and <https://www.crashkurs-statistik.de/multiple-lineare-regression/>

### Array.math LeftArray[], RightArray[], <operator\_sexp>, ResultArray[]

Operates the left array with the right array by the given operator string <operator\_sexp>.

For a command description see the corresponding BASIC! Functions.

The dimensions of the result array are equal to the **left** array.

The advantage if this command is the executing speed in opposite to For-Next loops.

The possible operators are:

`_+`, `_-`, `_*`, `_/`, `_min`, `_max`, `_atan2`, `_atan4`, `_hypot`, `_pow`, `_mod`, `_=`, `_<>`, `_<`, `_>`, `_<=`, `_>=`,  
`_shift`, `_bor`, `_band`, `_bxor`, `_huround`,  
`_:`, `_x3d`, `_x3d1`

To round in a Half-up mode use `_huround` for a fast execution.

The operator `_:` stands also for division, but to prevent infinity or Not a Number values `_:` has a Zero Check option. Instead of returning infinity or NaN value the divider is  $10^{-12}$ . If -0 is detectable the divider is  $-10^{-12}$ .

The cross product of 3d vectors will be returned by the `_x3d` operator.

If the array has more than three entries all three following entries will be interpreted as separate vectors.

The cross product with a vector length of 1 returns `_x3d1`. It is also Zero checked.

`_atan4` is special, because it returns the radiant for the term  $y / x$  in the range  $[0 \dots 2*PI]$

Example:

```
Array.load left[], 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 % 10 elements
Array.load right[], 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 % 10 elements
Array.math left[], right[], "_+", res[]
Print res[1] % Returns 0.0
Print res[5] % Returns 12.0
Array.math left[], right[], "_=", res[]
Print res[3] % Returns 1.0 for true
```

The following functions are assigned to the members of the **right** array.

**(op) can be +, -, \*, /, :**

`_(op)abs`, `_(op)sgn`, `_(op)ceil`, `_(op)floor`, `_(op)int`, `_(op)frac`, `_(op)round`,  
`_(op)const_pi`, `_(op)sin`, `_(op)sinh`, `_(op)asin`, `_(op)cos`, `_(op)cosh`, `_(op)acos`,  
`_(op)tan`, `_(op)tanh`, `_(op)atan`, `_(op)todegrees`, `_(op)toradians`,  
`_(op)sqr`, `_(op)cbrt`, `_(op)exp`, `_(op)log`, `_(op)log10`, `_(op)const_e`,  
`_(op)=0`, `_(op)<>0`, `_(op)bnot`, `_(op)even`, `_(op)odd`, `_(op)filter`,  
`_(op)2d1vect`, `_(op)3d1vect`

In case of `_(op)frac` and speed needs see also the description of the function `FRAC()`.

The vector length of 1 of 2d vectors will be returned by the `_(op)2d1vect` operator.

If the array has more than two entries all two following entries will be interpreted as separate vectors. It is also Zero checked.

The vector length of 1 of 3d vectors will be returned by the `_(op)3d1vect` operator.

If the array has more than three entries all three following entries will be interpreted as separate vectors. It is also Zero checked.

If you have no left array, fill in a dummy with zeros and proceed as follows:

Example:

```
Array.length al, RightArray[]
DIM LeftArray[al]
Array.fill LeftArray[], 0
! Return the content from RightArray[] as absolute values
Array.math LeftArray[], RightArray[], "_+abs", ResultArray[]
! Return the boolean values of the RightArray[] as 1.0 for true and 0.0 for false
! if the given value is not 0. Works like an on and off switch.
Array.math LeftArray[], RightArray[], "_+<>0", ResultArray[]
```

Example How to Scale a 3D Vector:

```
Array.load vector[], 1, 1, 4
s = 3
Array.load multi[], s, s, s
! Return the scaled vector. The result is in the vector array on the right side.
Array.math vector[], multi[], "_*", vector[] % Result is 3, 3, 12
```

Example Simple Linear Regression (regression line):

```
Array.Load x[], 156.3, 158.9, 160.8, 179.6, 156.6, ~
165.1, 165.9, 156.7, 167.8, 160.8 % KOERPERGROESSE
Array.Load y[], 47.1, 46.8, 49.3, 53.2, 47.7, 49.0, 50.6, 47.1, 51.7, 47.8 % RINGGROESSE
Array.average xAverage, x[] % Get the average of x[]
Array.average yAverage, y[]
Array.length al, x[] % Get the number of array entries
DIM xA[al], yA[al] % Create new arrays
Array.fill xA[], xAverage % Fill the array for later easy processing
Array.fill yA[], yAverage

Array.math x[], xA[], "_-", xD[] % Operates each x – x Average
Array.math y[], yA[], "_-", yD[]
Array.math xD[], yD[], "_*", yM[] % Multiplies each xD by each yD
Array.math xD[], xD[], "_*", yQ[] % Squares each xD

Array.sum z, yM[] % Get the sum of all array entries
Array.sum n, yQ[]
b = z / n % slope
a = yAverage - b * xAverage % intercept

array.min minX, x[]
array.max maxX, x[]
leftY = a + b * minX
rightY = a + b * maxX
PRINT "xAverage", xAverage
PRINT "yAverage", yAverage
PRINT "y = a + b * x"
PRINT "y = " + Str$(Round(a, 4)) + " + " + Str$(Round(b, 4)) + " * x"
PRINT "y (Xmin) = " + Str$(Round(leftY, 4))
PRINT "y (160) = " + Str$(Round(a + b * 160, 4))
PRINT "y (170) = " + Str$(Round(a + b * 170, 4))
PRINT "y (Xmax) = " + Str$(Round(rightY, 4))
```

Source:

Taken from <https://www.crashkurs-statistik.de/einfache-lineare-regression/#berechnen>

See also List.join, List.map.2d, List.map.3d, Array.Mat.Transpose



### **Array.max <Max\_nvar>, Array[{<start>,<length>}]**

Finds the maximum value in a numeric array (Array[]) or array segment (Array[start, length]), and places the result into the numeric variable <max\_nvar>.

### **Array.min <Min\_nvar>, Array[{<start>,<length>}]**

Finds the minimum value in a numeric array (Array[]) or array segment (Array[start, length]), and places the result into the numeric variable <min\_nvar>.

### **Array.median <Median\_nvar>, Array[]**

Returns the median of an array via <Median\_nvar>. The median is the value in the middle of the given but now temporarily sorted array. If the length of the specified array is even, the average of the two middle values is returned.

### **REDIM {<preserve\_nexp>}, Array[<nexp>{, <nexp> } ... ] ...**

### **REDIM {<preserve\_nexp>}, Array\$[<nexp>{, <nexp> } ... ] ...**

Re-Dimensions an existing array with optional 'preserve contents' flag.

Referenced arrays are not cut off (when passing to a user function).

Arrays must already exist before REDIM.

If the preserve flag is zero, the array will be full of 0's or "".

If the preserve flag is non-zero, the old contents will be preserved to the size of the old or new array, whichever was smallest. Any cells left-over will be 0's or "".

Preserve flags may be inserted anywhere in the arguments list.

Arrays following a preserve flag will honor that preserve flag until the next preserve flag.

Preserve flags cannot be array expressions e.g. p[3], as this would re-dim p[3].

Example:

REDIM 1, a[], 0, b[] 1, c[], d[]     % b[] will not be preserved

## Array Enhancements

### Array Assignments

OliBasic accepts Array pointers e.g `a[]`, `a$[]` for assignment and evaluation.

Assignments e.g `a[] = b[]`

`a[] = b[]` % makes an exact copy of `b[]` to `a[]` including the size and dimensions.

% background references (i.e passed by referenced variables) are preserved.

Evaluation (for future mod).

Evaluation of `a[]` always returns 0 ( might be changed in the future )

Evaluation of `a$[]` always returns "" ( might be changed in the future )

Evaluation of non-existent `a[]` also returns 0 or "" ( might be changed in the future )

The term 'array pointer' is a reference to an array. You may think of it as representing the whole array. This is inline with traditional Basic 01.91 which already uses the syntax for some commands and functions.

The behavior for array pointers (e.g `a[]`) has only been changed for evaluation and assignment.

It has not been changed when given to Basic commands or functions e.g `myfunc a[]`). This is because Basic commands and functions do their own parsing and have their own rules for arrays without indices. Re-Dimensions an existing array with optional 'preserve contents' flag.

## Filters

### **Filter.fft Real[], Imag[]**

Computes the discrete Fourier transform (DFT) of the given complex vector, storing the result back into the vector. The vector can have any length.

If the vector length has the power of 2, the Radix 2 algorithm is used else the more complicated Bluestein algorithm runs for any length.

A vector is a one-dimensional array.

A complex vector has two components, a real part Real[] and an imaginary part Imag[].

If you need the start vector again, copy it before using this command.

### **Filter.ifft Real[], Imag[]**

Computes the **inverse** discrete Fourier transform (IDFT) of the given complex vector, storing the result back into the vector. This transform does not perform scaling, so the inverse is not a true inverse. The vector can have any length.

If the vector length has the power of 2, the Radix 2 algorithm is used else the more complicated Bluestein algorithm runs for any length.

A vector is a one-dimensional array.

A complex vector has two components, a real part Real[] and an imaginary part Imag[].

If you need the start vector again, copy it before using this command.

### **Filter.circular.convolution.real xVec[], yVec[], outVec[]**

Computes the circular convolution of the given real vectors. Each vector's length must be the same.

See also [https://en.wikipedia.org/wiki/Circular\\_convolution](https://en.wikipedia.org/wiki/Circular_convolution)

### **Filter.circular.convolution.imag xReal[], xImag[], yReal, yImag[], outReal[], outImag[]**

Computes the circular convolution of the given complex vectors. Each vector's length must be the same.

See also [https://en.wikipedia.org/wiki/Circular\\_convolution](https://en.wikipedia.org/wiki/Circular_convolution)

### **Filter(<nexp>)**

Returns the filter result based on <nexp>. A filter has to be selected (Filter.set) beforehand.

This function can also be used by the command Array.math with the option `_(op)filter`.

Returns NaN if something went wrong.

**Filter.set <filter\_bundle>**

Selects the filter to be used next. The filter will be defined by the bundle <filter\_bundle>. Initial values will be set to the default or predefined ones.

Table of Bundle Keys		
Key	Value	Description
<b>_Filter</b>	_Average _Median	Set the filter type. <b>Always needed!</b>
	_Bessel _Butterworth _Chebyshev0_5dB _Chebyshev1_0db _Chebyshev2_0dB _Chebyshev3_0db _SinglePol1st _SinglePol2nd (String)	
<b>_Average Filter</b>		
Returns the average of given First-In-First-Out (FIFO) Array values.		
<b>_Length</b>	(numeric)	Limits the length of the FIFO array. The first-in values are not removed until the array length is greater than defined by _Length. Default is 5.
<b>_Section</b>	Array (numeric)	Sets a default initial array section. The length of FIFO Array is also given by length of the initial array section. The length will be only overwritten, if _Length is greater than the initial array section length.
<b>_Median Filter</b>		
Returns the median of given First-In-First-Out Array values. See the command Array.median for more details.		
<b>_Length</b>	(numeric)	Limits the length of the FIFO array. The first-in values are not removed until the array length is greater than defined by _Length. Default is 5.
<b>_Section</b>	Array (numeric)	Sets a default initial array section. The length of FIFO Array is also given by length of the initial array section. The length will be only overwritten, if _Length is greater than the initial array section length.

**For the next filters see also**

- [https://www.electronics-tutorials.ws/filter/filter\\_1.html](https://www.electronics-tutorials.ws/filter/filter_1.html) English
- <https://www.electronics-tutorials.ws/de/filtern/kapazitive-reaktanz.html> German

Note that waves can be manipulated also by mechanical devices like the combination of springs and dampers as an example. The exhaust tract of a car is nowadays usually also a sound filter.

The outputs have an attenuation of 3 dB.

**\_Bessel Filter**

A filter often used for sensors.

**\_Butterworth Filter**

A modern practical application of the filter is common in computer animation; it serves to reduce curve points without changing the general shape of the curve.

(Source: Wikipedia)

**\_Chebyshev0.5dB (also written as Tschhebyscheff) Filter**

The characteristic of the Butterworth filter is that the transfer characteristics in the stopband and passband are smooth. In the case of the Chebyshev filter, a ripple in the transmission characteristics is allowed in order to increase the steepness of the filter.

(Source: Hochschule Karlsruhe)

**\_Chebyshev1.0dB Filter**

**\_Chebyshev2.0dB Filter**

**\_Chebyshev3.0dB Filter**

**\_FirstOrder or One-Pole Filter**

Like a combination of a resistor and a capacitor or a damper and a spring.

**\_SecondOrder or Two-Pole Filter**

A combination of more different parts and may also an amplifier. Filters with higher orders are a combination of First Order and Second Order Filters.

**Advanced options for the filters from \_Bessel until \_SinglePol2nd**

<b>_InitialValue</b>	(numeric)	Sets an initial value. Default is 0.
<b>_CutOffRate</b>	(numeric)	Sets the cut off rate Nc. Tc: cut-Off time intervall Ts: sample time interval fc: cut-off frequency fs: sample frequency $Nc = Tc / Ts = fs / fc$ Default is 1.
<b>_PassType</b>	_Low or _High (String)	Sets the filter types lowpass filter or highpass filter. Default is _Low.

**Keep care for the next options, that only one option is active.**

The setting and effectiveness of logical filters is always a matter of trial and error and of course depends largely on the composition of the signal and the structure of the outliers.

<b>_AbsoluteError</b> (Option)	0 or 1 (numeric)	<p>A removal for absolute value outliers.</p> <p>For example, imagine a water tank with a water level sensor. The pump can only deliver a limited flow rate. The fill level in the measuring cycle can only move up by 3 cm (_UpperAbsolute). When fully open, the drain can lower the level by a maximum of 5 cm (_LowerAbsolute) per measurement cycle. If the absolutely possible change in value is determined in one of these cases, the last calculated output value is summed with the respective maximum possible absolute change in value for further calculation. To return a useful result the following absolute error values have to be set. Default is 0.</p>
<b>_UpperAbsolute</b>	(numeric)	<p>The upper value _UpperAbsolute for _AbsoluteError. Default is 0.</p>
<b>_LowerAbsolute</b>	(numeric)	<p>The lower value _LowerAbsolute for _AbsoluteError. Default is 0.</p>
<b>_Limits</b> (Option)	0 or 1 (numeric)	<p>A removal for limit outliers.</p> <p>This means e.g. in the case of a water level report from a rain barrel, the level cannot be lower than the bottom of the barrel (_LowerLimit) and not higher than the upper edge of the barrel (_UpperLimit). If these limits are exceeded, the last calculated output value is used for further calculation. To return a useful result the following limits have to be set. Default is 0.</p>
<b>_UpperLimit</b>	(numeric)	<p>Sets the upper value _UpperLimit for _Limits. Default is 0.</p>
<b>_LowerLimit</b>	(numeric)	<p>Sets the lower value _LowerLimit for _Limits. Default is 0.</p>

<b>_Damper</b>	> 0 (numeric)	The difference between the input value and the last output value is only accepted as 1/5 in the case of _Damper = 5, which additionally dampens a sudden increase in the output value in the event of outliers. In case of _Damper = 1 no damper is used. If _Damper < 1 and > 0, the opposite is true, that means the influence is amplified. Which is usually not desirable. Default is 1.
<b>_Descending (Option)</b>	0 or 1 (numeric)	This is a so-called "drag pointer filter". The output rises to the input value without delay and then slowly falls (descending) again with a delay. For example, you can use this in the case of an awning to react to wind events. If there are gusts, you have to react immediately (retract the awning), but the awning is not extended again with a time delay when the wind drops. Default is 0.

## Mesh commands

**Mesh.hull <result\_xyList\_nexp>, <x(y)List\_nexp>{, <yList\_nexp>}**

Returns a xy List of a convex hull polygon around a 2D point cloud.

The first List defined by <x(y)List\_nexp> can be contain x or xy values.

The optional y List defined by <yList\_nexp> contain only y values.

If the y list is not given, the first List has to be a xy List.

The result can be used directly by the Gr.poly command.

Example:

```
LIST.CREATE n, re % xy List
LIST.CREATE n, pl
LIST.ADD re, -300, 200, 100, 200, 100, -400, -300, -400 , 400, 500, 100, 300
MESH.HULL pl, re
GR.OPEN "_LightGreen", 1, 1 % Background
GR.COLOR "_Orange", 1 % Orgin
GR.CIRCLE oPtr0, 500, 500, 20

GR.COLOR "_Red", 0 % The hull polygon
GR.POLY oPtr2, pl, 500, 500

GR.COLOR "_Magenta", 1 % All points
LIST.SIZE re, sz
FOR i = 1 TO sz STEP 2
  LIST.GET re, i, x
  LIST.GET re, i + 1, y
  GR.CIRCLE oPtr, 500 + x, 500 + y, 10
NEXT
GR.COLOR "_Blue", 1 % Points along the hull polygon
LIST.SIZE pl, sz
FOR i = 1 TO sz STEP 2
  LIST.GET pl, i, x
  LIST.GET pl, i + 1, y
  GR.CIRCLE oPtr, 500 + x, 500 + y, 10
NEXT
GR.RENDER

DO
  PAUSE 100
UNTIL 0
```



**Mesh.stl.load** <triangles\_xyzList\_nexp>, <midpoints\_xyzList\_nexp>, <normals\_xyzList\_nexp>, {<header\_sval>}, <filePath\_sexp>{, <normal\_length\_nexp>}

Returns the xyz Lists <triangles\_xyzList\_nexp>, <midpoints\_xyzList\_nexp> and <normals\_xyzList\_nexp> from the file named by <filePath\_sexp>. The format is STL binary. The precision is 32 bit float ([https://en.wikipedia.org/wiki/STL\\_\(file\\_format\)](https://en.wikipedia.org/wiki/STL_(file_format))). The optional <header\_sval> returns the header of the binary STL file as an ASCII string in a length of 80 characters. If an error occurs the string of <header\_sval> starts with "\_Error". The absolute length of the plain normal vectors can be defined by <normal\_length\_nexp>. Default is no change. If the file specifies more triangles as with the file size possible, the command tries to load so many triangles as possible.

See also Mesh.triangle, Mesh.triangle.midpoint, Mesh.triangle.2.5d

**Mesh.stl.save** <triangles\_xyzList\_nexp>, <normals\_xyzList\_nexp>, <filePath\_sexp>{, <header\_sexp>}, <normal\_length\_nexp>}

Saves the xyz Lists <triangles\_xyzList\_nexp> and <normals\_xyzList\_nexp> into the file named by <filePath\_sexp>. The format is STL binary. The precision is 32 bit float ([https://en.wikipedia.org/wiki/STL\\_\(file\\_format\)](https://en.wikipedia.org/wiki/STL_(file_format))). The optional <header\_sexp> sets the header of the binary STL file as an ASCII string of maximal 80 bytes length. Longer ASCII strings will be cut off. Characters in the default UTF-8 format will be write as an ASCII character plus a question mark. The absolute length of the plain normal vectors can be defined by <normal\_length\_nexp>. Default is no change.

See also Mesh.triangle, Mesh.triangle.midpoint, Mesh.triangle.2.5d

**Mesh.triangle** <result\_xyList\_nexp>, <x(y)List\_nexp>{, <yList\_nexp>}

Returns a xy List of triangle polygons within a 2D point cloud using the Delaunay triangulation. All triangle points are in clockwise order if you use screen coordinates. Using the right hand order the points are sorted in the counterclockwise direction. If you use **World Coordinates** respectively the **Right-Hand Rule**, also known as a mathematically positive, the points are sorted in the counterclockwise direction. The first input List defined by <x(y)List\_nexp> can be contain x or xy values. The optional y List defined by <yList\_nexp> contain only y values. If the y List is not given, the first List has to be a xy List. Keep care that you do not use points with an equal xy coordinate, because this circumstance will be not checked by this command. The result can be used directly by the Gr.poly command.

**Mesh.triangle.midpoint** <result\_xyList\_nexp>, <x(y)List\_nexp>{, <yList\_nexp>}}

Returns a xy List of midpoints of triangle polygons within a 2D point cloud using the Delaunay triangulation.

The first input List defined by <x(y)List\_nexp> can be contain x or xy values.

The optional y List defined by <yList\_nexp> contain only y values.

If the y List is not given, the first List has to be a xy List.

Keep care that you do not use points with an equal xy coordinate, because this circumstance will be not checked by this command.

The result can be used to refine (triple) the mesh by combining this result with the input points.

**Mesh.triangle.2.5d** <xyzTriangles\_nexp>, <xyzMidpoints\_nexp>,  
<xyzNormals\_nexp>, {<normal\_length\_nexp>}, <xyzSource\_nexp>{,{  
<xyzOuterBorder\_nexp>}, <xyzInnerBorders\_nexp>}

**Returns** in <xyzTriangles\_nexp> a two and a half dimensional xyz List of **triangle polygons** within a 2D xy point cloud using the Delaunay triangulation. All triangle points are in clockwise order if you use screen coordinates.

Using the right hand order the points are sorted in the counterclockwise direction.

If you use **World Coordinates** respectively the **Right-Hand Rule**, also known as a mathematically positive, the points are sorted in the counterclockwise direction.

The z components are added afterwards.

**Returns** also in <xyzMidpoints\_nexp> a xyz List of **midpoints** of the triangle polygons. The result can be used to refine (triple) the mesh by combining this result with the input points.

The List <xyzNormals\_nexp> **returns** the **plain normal vectors** of the triangles.

The length of these vectors is specified by the optional <normal\_length\_nexp>.

If it is 0, an empty List will be returned. Default is 1. Often the coordinates of plane normals are denoted as i, j, k instead of x, y, z.

The **input** List is defined by <xyzSource\_nexp> containing the xyz values.

Keep care that you do not use points with an equal xy coordinate, because this circumstance will be not checked by this command. But it is more robust, because if it checks a 0, 0, 0 normal it blocks this triangle and use the first z value of this coordinate. But there is no guarantee that all possibilities are covered.

If your **outer border** of your mesh is not only convex, the optional xyz List

<xyzOuterBorder\_nexp> can be use to delete unwanted triangles outside the outer border.

Triangles of **inner contours** can be removed by a Bundle <xyzInnerBorders\_nexp>

containing Lists of xyz points which describe the inner polygons. Intersecting polygons give undesirable results.

Example:

```
List.create n, xyzPoints, xyzBorder
```

```
List.add xyzPoints, 0,0,0, 250,100,0, 500,0,0, 500,500,0, 250,400,0, 0,500,0
```

```
List.add xyzBorder, xyzPoints
```

```
List.add xyzPoints, 250,250,125
```

```
List.create n, triangles, midpoints, normals
```

```
Mesh.triangle.2.5d triangles, midpoints, normals, 1, xyzPoints , xyzBorder
```

## Functions

**Fn.def name|name\$( {nvar}|{svar}|Array[]|Array\$[], ... {nvar}|{svar}|Array[]|Array\$[]){[]}**

Begins the definition of a function. This command names the function and lists the parameters, if any.

If the function name ends with the \$ character then the function will return a string, otherwise it will return a number. **Ends the command with [], the function returns a string array or numeric array.** The parameter list can contain as many parameters as needed, or none at all. The parameters may be numeric or string, scalar or array.

Your program must execute **Fn.def** before it tries to call the named function. Your program must not attempt to create more than one function with the same name, or the same function more than once. However, you may override a built-in function by defining your own function with the same name.

The following are all valid:

```
FN.DEF cut$(a$, left, right)
FN.DEF sum(a, b, c, d, e, f, g, h, i, j)
FN.DEF sort(v$[], direction)
FN.DEF pi() % Overrides built-in. You can make  $\pi = 3!$ 
```

Parameters create variables visible only inside the function. They can be used like other variables created inside the function (see Variable Scope, above).

There are two types of parameters: call by reference and call by value. Call by value means that the calling variable value (or expression) is copied into the called variable.

Changes made to the called variable within the function do not affect the value of the calling variable. Call by reference means that the calling variable value is changed if the called variable value is changed within the function.

Scalar (non-array) function variables can be either call by value or call by reference. Which type the variable will be depends upon how it is called. If the calling variable has the "&" character in front of it, then the variable is call by reference. If there is no "&" in front of the calling variable name then the variable is call by value.

```
FN.DEF test(a)
a = 9
FN.RTN a
FN.END

a = 1
PRINT test(a), a %will print: 9, 1
PRINT test(&a), a %will print: 9, 9
```

Array parameters are always call by reference.

```
FN.DEF test(a[])
a[1] = 9
FN.RTN a[1]
FN.END

DIM a[1]
a[1] = 1
PRINT test(a[]), a[1] % prints: 9, 9
```

Along with the function's return value, you can use parameters passed by reference to return information to a function's caller.

### **Fn.rtn** <sexp>|<nexp>{[]}

Causes the function to terminate execution and return the value of the return expression <sexp>|<nexp> or an array of those types. The return expression type, string, ~~or~~ number or array, must match the type of the function name. **Fn.rtn** statements may appear anywhere in the program that they are needed.

A function can return ~~only~~ a single scalar value. It can~~not~~ return an array ~~also~~. It cannot return a data structure (List, Stack, Bundle, or graphical object), but it can return a pointer to a data structure.

Note: You can also return information to a function's caller through parameters passed by reference.

### **Fn.end**

Ends the definition of a user-defined function. Every function definition must end with **Fn.end**.

When your function is running, executing the **Fn.end** statement causes the function to terminate and return a default value. If the function type is numeric then the default return value is 0.0. A string function returns the empty string (""). A string function returns an one-dimensional array with one element (containing 0.0 or "").

Note: Maybe in future empty arrays will be allowed.

## LIST Commands

### List.create N|S, <pointer\_nvar>{, <pointer\_nvar>}...

Creates a new, empty List of the type specified by the N or S parameter. A List of strings will be created if the parameter is **S**. A List of numbers will be created if the parameter is **N**. Do not put quotation marks around the N or S.

The pointer to the new List will be returned in the <pointer\_nvar> variable.

The newly created List is empty. The size returned for a newly created List is zero.

**Note, an existing List will be cleared and overwritten without warning!**

Optionally, this command accepts more than one list of the same type also.

### List.add.list <destination\_list\_pointer\_nexp>, <source\_list\_pointer\_nexp>{, <sub\_list\_start\_nexp>, <sub\_list\_end\_nexp>}

Appends the elements in the source list to the end of the destination list.

The two lists must be of the same type (string or numeric).

If wished a sub list can be added by <sub\_list\_start\_nexp> and <sub\_list\_end\_nexp>.

Is only <sub\_list\_start\_nexp> given the value with this index will be added.

### List.spread <listOfLists\_pointer\_nexp>, Array[{<start>, <length>}], <count\_nexp>{, <clear\_nexp>}

### List.spread <listOfLists\_pointer\_nexp>, Array\$[{<start>, <length>}], <count\_nexp>{, <clear\_nexp>}

Spreads a count of elements section by section from a given Array range into Lists defined by the List pointers from the List <listOfLists\_pointer\_nexp>. The optional <clear\_nexp> controls that the Lists are cleared (1) before adding the values. If it is 0 the values are only added. Default is 1. The Lists and the Array have to be from the same type.

### List.replace <pointer\_nexp>, <index\_nexp>, <sexp>|<nexp>{, <index\_nexp>, <sexp>|<nexp>}...

The List element specified by <index\_nexp> in the list pointed to by <pointer\_nexp> is replaced by the value of the string or numeric expression.

The index is one-based. The first element of the list is 1.

The replacement expression type (string or numeric) must match the list type.

This command can replace more than one value if wished.

### List.remove <pointer\_nexp>,<index\_nexp>{{, <start\_nexp>}, <end\_nexp>}

Removes the list element specified by <index\_nexp> from the list pointed to by <pointer\_nexp>.

The index is ones based. The first element of the list is 1.

As an option a range from <start\_nexp> to <end\_nexp> can be removed.

### List.get <pointer\_nexp>, <index\_nexp>, <var>{, <index\_nexp>, <var>}...

The list element specified by <index\_nexp> in the list pointed to by <pointer\_nexp> is returned in the specified string or numeric variable <var>.

The index is one-based. The first element of the list is 1.

The return element variable type must match the list type (string or numeric).

returns more than one value if wished now.

More than one value can be returned as an option.

Example:

```
List.size xyzPoints, ls
FOR u = 1 TO ls STEP 3
  List.get xyzPoints, u, x, ++u, y, ++u, z % ++u equals u = u + 1
  PRINT x, y, z
NEXT
```

### List.clear <pointer\_nexp>{, <pointer\_nvar>}...

Clears the list pointed to by the list pointer and sets the list's size to zero.

Optionally, this command accepts more than one list also.

### List.kill.last

Kills the last List of the internal Lists list. Lists are global. If you create a List within a function so you are able to kill this List before leaving the function.

### List.row.print <pointer\_nexp>{{{, <step\_nexp>}, <lineNum\_nexp>}, <result\_svar>}

Prints a list as rows into the console or String. The list step range is defined by <step\_nexp>, which returns the row. Default is 1. If <lineNum\_nexp> is greater than 0, line numbers are returned at the start of the row. Default is 1. Is <result\_svar> given optionally, the rows are transferred into a String with line feed endings ("\n"). In this case no console output is returned.

List.row.print xyzPoints, 5 returns a wrong result, have to be fixed

Example:

```
List.add xyzPoints, 0,0,0, 250,100,0, 500,0,0, 500,500,0, 250,400,0, 0,500,0
List.row.print xyzPoints, 3 % Returns rows with line numbers and x, y, z values
```



## Advanced LIST Commands for Advanced Users

The next few commands take more effort to learn how they work and where the benefits are. One of them is an up to 30 time increasing speed instead of normal loops.

List.join is very powerful, but also complex.

It is useful to put your special solutions in separate functions with a simpler interface.

### List.split {<left\_nexp>}, {<right\_nexp>}, <source\_nexp>, <by\_reg\_sexp> { {{, <start\_nexp>}, <end\_nexp>}, <add\_nexp>}

Splits the source list <source\_nexp> into two lists and place them into <left\_nexp> and <right\_nexp> by the regular expression <by\_reg\_sexp> item by item. If the right part does not exist, an empty string "" or 0.0 is returned.

With the list type (S or N) you control the type of your output. The type of the source list is detected automatically.

The <left\_nexp> and <right\_nexp> lists are optional, but you definitely need one.

The parameters <start\_nexp> and <end\_nexp> point to the range of the source list to work with. If no value is given the begin respectively the end is used.

The <add\_nexp> argument let you add (<add\_nexp> = 1) the results to the output lists.

If this argument is 0 (default) the output lists are cleared before execution.

Warning: A source list can not be an output list in the same command.

See also: REPLACE\$, SPLIT, SPLIT.ALL

Example:

```
ARRAY.LOAD in1[], 1,2,27,4,5,6,7,8,3,4,57,114,115
LIST.CREATE s, resLeft
LIST.CREATE n, source
LIST.ADD.ARRAY source, in1[]
! resLeft will be contain a list of numbers as Strings
LIST.SPLIT resLeft, , source, "dummy"
DEBUG.ON
DEBUG.DUMP.LIST resLeft
! And backwards again
LIST.SPLIT source, , resLeft, "dummy"
DEBUG.DUMP.LIST source
```

! If you deal with BigDecimal numbers, this makes also sense as

! a faster solution in opposite to BigD.int and BibD.frac.

```
LIST.CREATE s, resRight
```

**! The point needs in Regular Expressions a double backslash as a special char.**

```
LIST.SPLIT resLeft, resRight, source, "\\."
```

```
PRINT "Int:"
```

```
DEBUG.DUMP.LIST resLeft
```

```
PRINT "Frac:"
```

```
DEBUG.DUMP.LIST resRight
```



**List.split.2d <xList\_nexp>, <yList\_nexp>, <xySource\_nexp>{, <add\_nexp>}**

Splits a numeric **xy** source List <**xySource\_nexp**> into the returned x and y Lists. If the optional <add\_nexp> is greater than 0, the results will be added into the returned Lists otherwise the Lists will be cleared before using. Default is 0.

Note, all Lists have to be numeric.

Example:

```
List.split.2D xPoints, yPoints, xyPoints
```

**List.split.3d <x(y)List\_nexp>, {<yList\_nexp>}, {<zList\_nexp>}, <xyzSource\_nexp>{, <add\_nexp>}**

Splits a numeric **xyz** source List <**xyzSource\_nexp**> into the returned **x**, **y** and **z** Lists. If the optional <add\_nexp> is greater than 0, the results will be added into the returned Lists otherwise the Lists will be cleared before using. Default is 0.

If the optional <**yList\_nexp**> is not given, <**x(y)List\_nexp**> returns a **xy** List.

If the List <**zList\_nexp**> is optional.

Note, all Lists have to be numeric.

See also Gr.poly

Example 1:

```
List.split.3D xPoints, yPoints, zPoints, xyzPoints
```

Example 2:

```
List.split.3D xPoints, yPoints, , xyzPoints
```

Example 3:

```
List.split.3D xyPoints, , zPoints, xyzPoints
```

Example 4:

```
List.split.3D xyPoints, , , xyzPoints
```

**List.join <result\_nexp>, {<scr\_left\_nexp>|<scr\_left\_sexp>},  
{<scr\_right\_nexp>|<scr\_right\_sexp>}, <delim\_sexp> {{{, <\_oper\_arg\_sexp>},  
<start\_nexp>}, <end\_nexp>}, <add\_nexp>}**

Joins the optional source lists <scr\_left\_nexp> and <scr\_right\_nexp> into list <result\_nexp> item by item. The list <scr\_right\_nexp> is optional.

With your chosen list type (S or N) you control the type of your output <result\_nexp> automatically. The type of source lists is recognized and converted internally and automatically into strings.

The optional <scr\_left\_sexp> and <scr\_right\_sexp> are representing a list with limited size. The size limits depend on the size of the other list or the arguments <end\_nexp> and <add\_nexp>. Numeric values have to be converted to Strings maybe with STR\$(<nexp>).

Is the item count of the lists <scr\_left\_nexp> and <scr\_right\_nexp> different, the item count of the list with the most items or the setting is used. In this case the empty item returns "" or 0.0 ( \_+(.), \_-(.) ) or 1.0 ( \_\*(.), \_/(.)([0-9]) ). **Study with trying two lists with different length and learn with the results!** Compare with the example line which ends with **!\*/!**.

The delimiter <delim\_sexp> is added normally after the <scr\_left\_nexp> item. After that the optional <\_oper\_arg\_sexp> is executed before adding into the list <result\_nexp>.

The parameters <start\_nexp> and <end\_nexp> point to the range of the source list to work with. If no value is given the begin respectively the maximum end is used. Remember that <end\_nexp> is **not limited!** See the example line which ends with "A index list function".

The <add\_nexp> argument lets you add (<add\_nexp> = 1) the results to the output lists. If this argument is 0 (default) the output lists are cleared before execution.

## Operators

For String expressions valid operators are: \_+\$

For Numeric expressions valid operators are: \_+, \_-, \_\*, \_/{scale\_nexp},  
\_\*sin, \_\*cos, \_\*tan, \_\*asin, \_\*acos, \_\*atan, \_\*sqr

You can also put a point (.) into the operator to account for zeros (0) before missed decimal points like , \_+., \_-., \_\*., \_/.{scale\_nexp}, \_\*.sin ....

The optional {scale\_nexp} is set per default to 16.

\_+i equals the list index and \_-i equals the list index as a negative number

\_atan4 is special, because it returns the angle for the term y / x in the range [0° ... 360°] **counterclockwise**. <scr\_left\_nexp> is **y!!!**, <scr\_right\_nexp> is **x**.

Note, if y = 0 and x = 0, the result is 0.

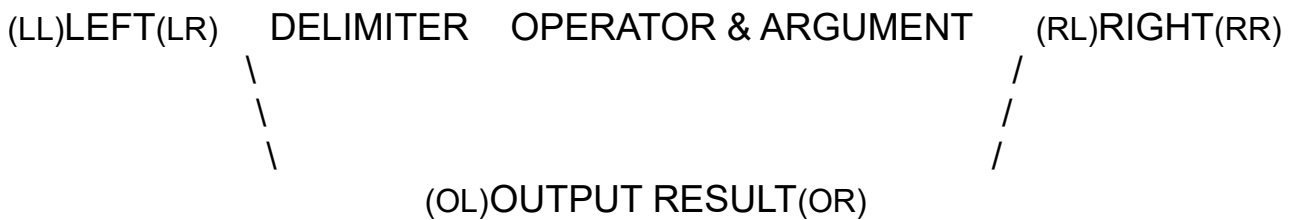
\_min, \_max are special, because it returns the minimum or maximum of <scr\_left\_nexp> and <scr\_right\_nexp> is **x**.

The four basic arithmetic operations are calculated as `BigDecimal`, after the entire input has been converted to strings before. Other operations will be computed as `Double`. Keep in mind, that values from type `Double` contain only maximal 15 **correct** digits. Trigonometric functions use **degrees** as in- and output.

If the `<_oper_arg_sexp>` starts additionally with `_D` all arithmetic operations are calculated as Double, that increases the speed round about 130% with lost of accuracy. In this case the scale argument has no effect.

## Arguments

Arguments as string expressions have to be enclosed in quotation marks "text" ( \"text\" or + CHR\$(34) + "text" + CHR\$(34) + ).



Examples:

```
a <scr_left_nexp> item = 20
a <scr_rigth_nexp> item = 010
<delim_sexp> = "." and <_oper_arg_sexp> = "" returns 20.010
<delim_sexp> = "" and <_oper_arg_sexp> = "+" returns 30
<delim_sexp> = "." and <_oper_arg_sexp> = "+" returns 20.1
<delim_sexp> = "" and <_oper_arg_sexp> = "+." returns 20.01
<delim_sexp> = "" and <_oper_arg_sexp> = "/.3" returns 2000.000
<delim_sexp> = "" and <_oper_arg_sexp> = "min." returns 0.01
<delim_sexp> = "$" and <_oper_arg_sexp> = "+$ "+#" returns #20$010#
```

```
only <scr_left_nexp> item = 20
<delim_sexp> = "" and <_oper_arg_sexp> = "_._04" returns 19.96
only <scr_rigth_nexp> item = 010
<delim_sexp> = "" and <_oper_arg_sexp> = "_.*_2.5" returns 0.025 % If left side is "", 1 is used !*/!
no source lists, <start_nexp> = 5, <end_nexp> = 6,
<delim_sexp> = "" and <_oper_arg_sexp> = "-i" returns -5 and -6 % A index list function
```

Warning: A source list can not be an output list in the same command.

Note: The special Floating Point numbers NaN and Infinity are not supported, if using BigDecimal.

### Example of Creating an Index List:

```
List.create n, indexList
iStart = 1
iEnd = 10
List.join indexList, , "", "_D_-i", iStart, iEnd % Returns from -1 to -10
List.add indexList, 0
List.join indexList, , "", "_D_+i", iStart, iEnd, 1 % Adds from +1 to +10
List.sort indexList, 0 % Sort mode ascending
List.row.print indexList, 1 % Shows 21 items from -10 to 10 in the console.
```

**List.join.2d** <xyList\_nexp>, <xSource\_nexp>, <ySource\_nexp>{, <add\_nexp>}

Joins the numeric x and y source Lists <xSource\_nexp> and <ySource\_nexp> into the returned **xy** List <xyList\_nexp>. If the optional <add\_nexp> is greater than 0, the result will be added into the returned List otherwise the List will be cleared before using.

Default is 0.

Note, all Lists have to be numeric.

Example:

List.join.2D xyPoints, xPoints, yPoints

**List.join.3d** <xyList\_nexp>, <x(y)Source\_nexp>, {<ySource\_nexp>},  
{<zSource\_nexp>}{, <add\_nexp>}, <preZ\_nexp>}

Joins the numeric x, y and z source Lists <x(y)Source\_nexp> and <ySource\_nexp>, <zSource\_nexp> into the returned **xyz** List <xyzList\_nexp>. If the optional <add\_nexp> is greater than 0, the result will be added into the returned Lists otherwise this List will be cleared before using. Default is 0.

If the optional <ySource\_nexp> is not given, <x(y)Source\_nexp> as a **xy** List is used.

If the optional <zSource\_nexp> is not given, predefined **z** values by the optional <preZ\_nexp> are inserted. If in this case <preZ\_nexp> not given the default value of 0.0 is inserted.

Note, all Lists have to be numeric.

Example 1:

```
List.join.3D xyzPoints, xPoints, yPoints, zPoints
```

Example 2:

```
List.join.3D xyzPoints, xyPoints, , zPoints
```

Example 3:

```
List.join.3D xyzPoints, xPoints, yPoints, , 0, preZ
```

Example 4:

```
List.join.3D xyzPoints, xyPoints, , , 0, preZ
```

**List.binary.search** <pointer\_nexp>, search\_nexp|search\_sexp, <result\_nvar>

Searches the specified list for the specified string or numeric value. The position of the first (left-most) occurrence is returned in the numeric variable <result\_nvar>. If the value is not found in the list then the result is zero.

This command use the very fast binary-search-method. **Keep in mind, that you have to sort the list before to prevent unexpected results.**

Example:

```
List.sort IPtr, 0, "fr_FR"
```

```
List.binary.search IPtr, "Paris", resultIndex
```

**List.match** {<index\_nexp>}, {<result\_nexp>}, <source\_nexp>,  
<by\_find\_sexp> {{{{, <start\_nexp>}, <end\_nexp>}, <add\_nexp>},  
<mode\_sexp>}, <inverse\_sexp>}

Checks with the expression <by\_find\_sexp> for matches in the source list <source\_nexp> and put the corresponding index into the optional list <index\_nexp> and the result into the optional list <result\_nexp>.

With your chosen list type (S or N) you control the type of your output <result\_nexp> automatically. The type of the source list is detected automatically.

The parameters <start\_nexp> and <end\_nexp> point to the area of the source list with which you want to work. If no value is given the begin respectively the end is used.

The <add\_nexp> argument let you add (<add\_nexp> = 1) the results to the output lists.

If this argument is 0 (default) the output lists are cleared before execution.

Options of matching mode <mode\_sexp>:

"\_RegEx" stands for regular expressions.

Valid **String expressions** are: \_Default = \_Is\_In, \_Is\_In\_IgnoreCase, \_Starts\_With, \_Starts\_With\_IgnoreCase, \_Ends\_With, \_Ends\_With\_IgnoreCase, \_Equals, \_Equals\_IgnoreCase, **\_RegEx\_Not**, \_RegEx\_First and \_RegEx\_First\_IgnoreCase.

Valid **Numeric expressions** are: \_Default = \_Is\_In\_IgnoreCase, \_Starts\_With\_IgnoreCase, \_Ends\_With\_IgnoreCase, \_Equals\_IgnoreCase, \_RegEx\_First\_IgnoreCase, \_=, \_<, \_<=, \_>, \_>= and \_Equals\_Numeric.

Options of the inverse mode <inverse\_sexp>: \_! , \_Not or an empty string(default).

The opposite of the matches will be returned.

Warning: A source list can not be an output list in the same command.

See also: REPLACE\$, SPLIT, SPLIT.ALL

**List.sort** <pointer\_nexp>{{{, <sort\_mode\_nexp>}, <locale\_sexp>},  
<strength\_sexp>}

Sorts the content of the given list.

Options: <sort\_mode\_nexp>:

- 0 Sorted in ascending, UTF-8 character table numbered order (default)
- 1 Sorted in descending, UTF-8 character table numbered order

If the <locale\_sexp> is set, the output is based on language and region. The locale specifies the language and region with standardized codes. The <locale\_sexp> is a string containing zero or more codes separated by underscores.

The function accepts up to three codes. The first must be a language code, such as "en", "de" or "ja".

The second must be a region or country code, such as "FR", "US", or "IN". Some language and country combinations can accept a third code, called the "variant code".

The function also accepts the standard three-letter codes and numeric codes for country or region. For example, "fr\_FR", "fr\_FRA", and "fr\_250" are all equivalent.

If <locale\_sexp> = "" , meaning "use my default locale".

If <locale\_sexp> = empty, the list is sorted by the order of the character map like

Array.sort.

To control the strength of the sorting use <strength\_sexp> with the keys described in List.sort.by.

**List.sort.by {<index\_nexp>}, {<toSort\_nexp>}, <by\_nexp>{{{, <sort\_mode\_nexp>}, <locale\_sexp>}, <strength\_sexp>}**

Creates an optional index list <index\_nexp> by sorting the content of the given list <by\_nexp>. Optional will be the list <toSort\_nexp> ordered by the internally created index. The list <index\_nexp> will be overwritten.

The lists <toSort\_nexp> and <by\_nexp> should be the same size and their types numeric and string can be combined.

Options: <sort\_mode\_nexp>:

- 0 Sorted in ascending, UTF-8 character table numbered order (default)
- 1 Sorted in descending, UTF-8 character table numbered order

If the <locale\_sexp> is set, the output is based on language and region. The locale specifies the language and region with standardized codes. The <locale\_sexp> is a string containing zero or more codes separated by underscores.

The function accepts up to three codes. The first must be a language code, such as "en", "de" or "ja".

The second must be a region or country code, such as "FR", "US", or "IN". Some language and country combinations can accept a third code, called the "variant code".

The function also accepts the standard three-letter codes and numeric codes for country or region. For example, "fr\_FR", "fr\_FRA", and "fr\_250" are all equivalent.

If <locale\_sexp> = "" , meaning "use my default locale".

If <locale\_sexp> = empty, the list is sorted by the order of the character map like Array.sort.

To control the strength of the sorting use <strength\_sexp> with the following keys.

"\_Primary". Typically, recognizes differences in the base character so that "a" is smaller than "b". There are no differences between accents and umlauts, so that "a", "ä" and "á" are the same.

"\_Secondary". **Is the default key.** Detects characters with accents. So "a" and "á" are not the same anymore as in \_Primary. Accents in the characters are considered secondary differences (for example, "as" < "às" < "at"). Other differences between letters can also be considered secondary differences, depending on the language. A secondary difference is ignored when there is a primary difference anywhere in the strings.

"\_Tertiary". Distinguishes in upper and lower case; in \_Primary and \_Secondary the spelling does not matter, and "a" is equal to "A". Upper and lower case differences in characters are distinguished at tertiary strength (for example, "ao" < "Ao" < "aò"). In addition, a variant of a letter differs from the base form on the tertiary strength (such as "A" and "Ⓐ"). Another example is the difference between large and small Kana. A tertiary difference is ignored when there is a primary or secondary difference anywhere in the strings.

"\_Identical". Really all Unicode characters are different. While the first three constants treat non-visible characters like CHR\$(1) or CHR\$(6) the same, they're really different under \_Identical. When all other strengths are equal, the \_Identical strength is used as a tiebreaker. For example, Hebrew cantillation marks are only distinguished at this strength. This strength should be used sparingly, as only code point value differences between two strings are an extremely rare occurrence. Using this strength substantially decreases the performance.

Examples for the "de" language code:

**\_Primary**

abc = ABC

Quäken = Quaken

boß = boss

boß < boxen

**\_Secondary**

abc = ABC

Quäken > Quaken

boß = boss

boß < boxen

**\_Tertiary**

abc < ABC

Quäken > Quaken

boß > boss

boß < boxen

Example for special German sortings:

DEBUG.ON

LIST.CREATE s, toSort, by

LIST.ADD toSort, "Goldmann", "Göbel", "Goethe", "Göthe", "Götz"

PRINT "List of German words to sort"

DEBUG.DUMP.LIST toSort

LIST.TOARRAY toSort, mem\$[] % Copy the content of the toSort list

LIST.ADD.ARRAY by, mem\$[] % into the by list.

PRINT "German DIN 5007 variant 1; used for words, such as in dictionaries"

LIST.SORT.BY , toSort, by, , "de", "\_Secondary"

DEBUG.DUMP.LIST toSort

LIST.SIZE by, IS

PRINT "German DIN 5007 variant 2; special sorting for name lists, ";

PRINT "for example in telephone directories"

FOR i = 1 TO IS

LIST.GET by, i, str\$

str\$ = LOWER\$(str\$)

str\$ = REPLACE\$(str\$, "ä", "ae")

str\$ = REPLACE\$(str\$, "ö", "oe")

str\$ = REPLACE\$(str\$, "ü", "ue")

LIST.REPLACE by, i, str\$

NEXT

LIST.CLEAR toSort % Refresh the toSort list

LIST.ADD.ARRAY toSort, mem\$[] % with the original content.

LIST.SORT.BY , toSort, by, , "de", "\_Secondary"

DEBUG.DUMP.LIST toSort



```

PRINT "Austrian telephone directory sorting"
LIST.CLEAR by % Refresh the toSort list
LIST.ADD.ARRAY by, mem$[] % with the original content.
LIST.SIZE by, IS
FOR i = 1 TO IS
  LIST.GET by, i, str$
  str$ = LOWER$(str$)
  str$ = REPLACE$(str$, "ä", "az")
  str$ = REPLACE$(str$, "ö", "oz")
  str$ = REPLACE$(str$, "ü", "uz")
  str$ = REPLACE$(str$, "ß", "ssz")
  str$ = REPLACE$(str$, "st.", "sankt")
  LIST.REPLACE by, i, str$
NEXT
LIST.CLEAR toSort
LIST.ADD.ARRAY toSort, mem$[]
LIST.SORT.BY , toSort, by, , "de", "_Secondary"
DEBUG.DUMP.LIST toSort

```

Example how to use a sorting index:

```

ARRAY.LOAD x[], 10, 2, 9, 7, -8, 99, 4, 7, 1, 4, -13
ARRAY.LOAD y[], -8, 99, 4, 7, 1, 4, -13, 10, 2, 9, 7
ARRAY.LOAD z[], 10, 2, 9, 1, 4, -13, 7, -8, 99, 4, 7
PRINT "In this case the source data have not to be copied or changed."
PRINT " X", " Y", " Z", " sorted by Y"
LIST.CREATE n, byY, idxList
LIST.ADD.ARRAY byY, y[]
LIST.SORT.BY idxList, , byY
LIST.TOARRAY idxList, idx[]
ARRAY.LENGTH aL, idx[]
FOR i = 1 TO aL
  IF i = 1 THEN PRINT "The min: ";
  IF i = aL THEN PRINT "The max: ";
  PRINT x[idx[i]], y[idx[i]], z[idx[i]]
NEXT

```

**List.dimsort.by** <sorted\_nexp>, <toSort\_nexp>, <dim\_s\_nexp>, <by\_nexp>, <dim\_b\_nexp>, <which\_nexp>{{, <sort\_mode\_nexp>}, <exclude\_sexp>, <exValue\_sexp>}

The returned numeric List <sorted\_nexp> is a copy of the numeric List <toSort\_nexp>, but it is ordered by the internally sorting index. Its <dim\_s\_nexp> specifies the length of the entry sets, which are sorted. The internal sorting index is created by the numeric list <by\_nexp>. Its <dim\_b\_nexp> specifies the length of the entry sets that contain the entry that forms the basis of the sort. The argument <which\_nexp> specifies the member of each set which is the basis.

Options of <sort\_mode\_nexp>:

- -1 Nothing is sorted, but a copy under conditions is optional possible.
- 0 Sorted in ascending order (default)
- 1 Sorted in descending order

If the optional <exclude\_sexp> specifies a condition, under which it is possible to exclude an entry set. Conditions are `_ =`, `_ <`, `_ <`, `_ >`, `_ <=`, `_ >=` as Strings. The value to be compared to is given by <exValue\_sexp>.

Example:

```
List.create n, xy, theResult
List.add xy, 15, 5, 44, 4, 23, 3, 12, 2, 14, 1 % Two dimensional xy List
List.dimsort.by theResult, xy, 2, xy, 2, 2, 0, "_<", 5
! The List theResult returns 14, 1, 12, 2, 23, 3, 44, 4 sorted by the y entries.
List.row.print theResult, 2
```

**List.bounds.2d** <pointer\_nexp>, <xMin\_nvar>, <yMin\_nvar>, <xMax\_nvar>, <yMax\_nvar>

Gets the bounding rectangle of an xy list that can be used in conjunction with gr.polygon. For a complete 2D operation you need the x and the y value, in this consequence an even number of values.

Is the number of the list items not even, the last value will be ignored.

Example:

```
LIST.CREATE n, ln1
LIST.ADD ln1, 100, 100, 200, 100
LIST.ADD ln1, 200, 200, 100, 200
```

```
mvX = 0: mvY = 0 % Move x,y for testing
LIST.CREATE n, ln2
LIST.ADD ln2, mvX + 60, mvY + 150, mvX + 150, mvY + 60
LIST.ADD ln2, mvX + 240, mvY + 150, mvX + 150, mvY + 240
```

```
LIST.BOUNDS.2D ln1, left, top, right, bottom
PRINT "ln1:", left, top, right, bottom
ARRAY.LOAD r1[], left, top, right, bottom % Result 1
```

```
LIST.BOUNDS.2D ln2, left, top, right, bottom
PRINT "ln2:", left, top, right, bottom
ARRAY.LOAD r2[], left, top, right, bottom % Result 2
```

```
left = 1: top = 2: right = 3: bottom = 4
collision = 1
IF ((r1[bottom] < r2[top]) | (r2[bottom] < r1[top]) | (r1[right] < r2[left]) | (r2[right] < r1[left])) ~
    THEN collision = 0 % Test for collision
PRINT "collision: "; collision
```

```
PAUSE 3000
```

```
GR.OPEN "_White", 1, 1
GR.COLOR "_Red"
GR.POLY obj1, ln1, 0, 0
GR.COLOR "_Green", 0
GR.POLY obj2, ln2, 0, 0
GR.RENDER
```

```
DO
    PAUSE 100
UNTIL 0
```

**List.bounds.3d** <pointer\_nexp>, <xMin\_nvar>, <yMin\_nvar>, <zMin\_nvar>, <xMax\_nvar>, <yMax\_nvar>, <zMax\_nvar>

Gets the bounding box of a xyz list.

For a complete 3D operation you need the x, y and the z value, in this consequence the division from the number of list items by 3 has to be an integer. Is a 3D vector not complete, it will be ignored.

```
List.map.2d <pointer_nexp>, {{{{{{{{{dx1},dy1,agl1},dx2},dy2,agl2},mulx},muly}
```

All Arguments are of the type <nexp>.

Maps results of 2D operations by translation 1, rotation 1, translation 2, rotation 2 and multiplication (in this order) back into a given **x/y** value list.

This covers different cases of a combined workflow, which is often needed.

The arguments for translation are dx1, dy1, dx2 and dy2 (mostly in longitudinal units).

The arguments for rotation are aql1 and aql2 in degrees.

The arguments for multiplication are `mulx` and `muly`.

The default arguments for translation and rotation are set to 0.

The default arguments for multiplication are set to 1.

For a complete 2D operation you need the x and the y value, in this consequence an even number of values.

Is the number of the list items not even, the last value is only computed by dx1, dx2 and mulx. So you can use also 1D value lists if the x/y arguments are equal and agl1 respectively aql2 are zero.

If you use **Screen Coordinates** you get a **clockwise** rotation.

If you use **World Coordinates** respectively the **Right-Hand Rule** you get **counter-clockwise** rotation or known as a mathematically positive rotation direction.

Example:

List.create n, l2d

```
List.add l2d, 0, 1, 1, 1, 7 %The number of list items is not even!
```

```
List.map.2d l2d, 0,0, 45, 0,0, 0, 2,2
```

Debug.on

## Debug.dump.list l2d % Look what happens with the 7.

See also [Array.Mat.Transpose](#), [Array.Math](#)

**List.map.3d** <pointer nexp>.

```
{{{{{{{{{{{{{{dx1},dy1},dz1},agl1x},agl1y},agl1z},{dx2},{dy2},{dz2},agl2x},agl2y},agl2z},mulx},muly},mulz}
```

All Arguments are of the type <nexp>.

Maps results of 3D operations by translation 1, rotation 1, translation 2, rotation 2 and multiplication (in this order) back into a given **x/y/z** value list.

This covers different cases of a combined workflow, which is often needed.

The arguments for translation are dx1, dy1,dz1, dx2, dy2 and dz2 (mostly in longitudinal units).

The arguments for rotation are agl1x, agl1y, agl1z, agl2x , agl2y and agl2z in degrees.

The arguments for multiplication are `mulx`, `muly` and `mulz`.

The default arguments for translation and rotation are set to 0.

The default arguments for multiplication are set to 1.

For a complete 3D operation you need the x, y and the z value, in this consequence the division from the number of list items by 3 has to be an integer.

If the list has one or two items more, this items are in **opposite** to List.map.2d **not changed**.

If you use **Screen Coordinates** you get a **clockwise** rotation.

If you use **World Coordinates** respectively the **Right-Hand Rule** you get **counter-clockwise** rotation or known as a mathematically positive rotation direction.

See also [Array.Mat.Transpose](#), [Array.Math](#)

Example:

```
List.create n, l3d
List.add l3d, 0, 1, 0, 1, 1, 0, 7 % The division of the number of list items
                                % by 3 is not an integer.
Array.load t3d1[], 0,0,0 % Translation 1 [x,y,z]
Array.load r3a1[], 0,0,45 % Rotation 1 in degrees around [x,y,z]-Axis
Array.load t3d2[], 0,0,0 % Translation 2 [x,y,z]
Array.load r3a2[], 0,0,0 % Rotation 2 in degrees around [x,y,z]-Axis
Array.load m3d[], 1,1,1 % Multiplication [x,y,z]
List.map.3d l3d, t3d1[1],t3d1[2], t3d1[3], r3a1[1], r3a1[2], r3a1[3], ~
t3d2[1],t3d2[2], t3d2[3], r3a2[1], r3a2[2], r3a2[3], ~
m3d[1], m3d[2], m3d[3]
Debug.on
Debug.dump.list l3d % Look what happens with the 7.
```

**List.replace.by <pointer\_nexp>, <index\_pointers\_nexp>, <value\_pointer\_nexp>**

**List.replace.with <pointer\_nexp>, <index\_pointers\_nexp>, <value\_pointer\_nexp>**

**List.replace.boolean <pointer\_nexp>, <booleans\_nexp>, <value\_pointer\_nexp>**

## Stacks

### **Stack.kill.last**

Kills the last Stack of the internal Stacks list. Stacks are global. If you create a Stack within a function so you are able to kill this Stack before leaving the function.

## SQLITE Command Enhancements

Also good sources are at <http://www.sqlitetutorial.net> and <https://www.sqlite.org/faq.html#q2>.

**Sql.new\_table <DB\_pointer\_nvar>, <table\_name\_sexp>, C1\$, C2\$, ...,CN\$**

**Sql.new\_table <DB\_pointer\_nvar>, <table\_name\_sexp>, <delim\_row\_sexp>**

A single database may contain many tables. A table is made of rows of data. A row of data consists of columns of values. Each value column has a column name associated with it.

This command creates a new table with the name <table\_name\_sexp> in the referenced opened database, **but only if the table does not exist**. The column names for that table are defined by the following: C1\$, C2\$, ..., CN\$. At least one column name is required. You may create as many column names as you need.

BASIC! always adds a Row Index Column named "\_id" to every table. The value in this Row Index Column is automatically incremented by one for each new row inserted. This gives each row in the table a unique identifier. This identifier can be used to connect information in one table to another table. For example, the \_id value for customer information in a customer table can be used to link specific orders to specific customers in an outstanding order database.

The alternative <delim\_row\_sexp> is a string expression starting with "\_Delimiter:" + one delimiter character as a numeric string + ";" + C1\$ + dC\$ + C1\$ + ... + dC\$ + CN\$

Using other delimiter characters is a little advantage. It is possible to program the number of columns.

Example:

```
delimN = ucode("€")
```

```
dC$ = chr$(delimN)
```

```
head$ = "_Delimiter:" + int$(delimN) + ";"
```

```
columns$ = head$ + c1$ + dC$ + c2$ + dC$ + c3$ + dC$ + c4$
```

```
SQL.NEW_TABLE DB_Ptr, tbname$, columns$
```

**Sql.drop\_table <DB\_pointer\_nvar>, <table\_name\_sexp>**

The table named <table\_name\_sexp> in the opened database pointed to by <DB\_pointer\_nvar> will be dropped (**deleted**) from the database if the table exists.

**Sql.insert <DB\_poinnameter\_nvar>, <table\_\_sexp>, C1\$, V1\$, C2\$, V2\$, ..., CN\$, VN\$**

**Sql.insert <DB\_pointer\_nvar>, <table\_name\_sexp>, <delim\_row\_sexp>**

Inserts a new row of data columns and values into a table in a previously opened database.

The <table\_name\_sexp> is the name of the table into which the data is to be inserted. All newly inserted rows are inserted after the last, existing row of the table.

C1\$, V1\$, C2\$, V2\$, ..., CN\$, VN\$: The column name and value pairs for the new row.

These parameters must be in pairs. The column names must match the column names

used to create the table. Note that the values are all strings. When you need a numeric

value for a column, use the BASIC! STR\$(n) to convert the number into a string. You can

also use the BASIC! FORMAT\$(pattern\$, N) to create a formatted number for a value.

(The Values-as-strings requirement is a BASIC! SQL Interface requirement, not a SQLite requirement. While SQLite, itself, stores all values as strings, it provides transparent conversions to other data types. I have chosen not to complicate the interface with access to these SQLite conversions since BASIC! provides its own conversion capabilities.)

The alternative <delim\_row\_sexp> is a string expression starting with "\_Delimiter:" + one delimiter character as a numeric string + ";" + C1\$ + dC\$ + V1\$ + ..., CN\$ + dC\$ + VN\$

Using other delimiter characters is a little advantage. It is possible to program the number of columns.

Example:

```
delimN = ucode("€")
```

```
dC$ = chr$(delimN)
```

```
head$ = "_Delimiter:" + int$(delimN) + ";"
```

```
row$ = head$ + c1$ + dC$ + fn$ + dC$ + c2$ + dC$ + ln$ + dC$ + c3$ + dC$ + ga$ + ~
```

```
dC$ + c4$ + dC$ + pa$
```

```
SQL.INSERT DB_Ptr, tbname$, row$
```



**Sql.update** <DB\_pointer\_nvar>, <table\_name\_sexp>, C1\$, V1\$, C2\$, V2\$,...,CN\$, VN\$  
{: <where\_sexp>}

**Sql.update** <DB\_pointer\_nvar>, <table\_name\_sexp>, <delim\_row\_sexp> {:  
<where\_sexp>}

In the named table of a previously opened database, change column values in specific rows selected by the Where\$ parameter <where\_sexp>. The C\$,V\$ parameters must be in pairs. The colon character terminates the C\$,V\$ list and must precede the Where\$ in this command. The Where\$ parameter and preceding colon are optional.

BASIC! also uses the colon character to separate multiple commands on a single line. The use of a colon in this command conflicts with that feature. Use caution when using both together.

If you put a colon on a line after this command, the preprocessor **always** assumes the colon is part of the command and not a command separator. If you are not certain of the outcome, the safest action is to put the **Sql.update** command on a line by itself, or at the end of a multi-command line.

The alternative <delim\_row\_sexp> is a string expression starting with "\_Delimiter:" + one delimiter character as a numeric string + ";" + C1\$ + dC\$ + V1\$ + ..., CN\$ + dC\$ + VN\$ Using other delimiter characters is a little advantage and programming the number of the parameter in pairs is possible.

Example:

```
delimN = ucode("€")
```

```
dC$ = chr$(delimN)
```

```
head$ = "_Delimiter:" + int$(delimN) + ";"
```

```
Where$ = "first_name = 'Tamasin' AND last_name = 'Washington' "
```

```
! SQL.UPDATE DB_Ptr, tbnam$, c3$, "94": Where$ or
```

```
row$= head$ + c3$ + de$ + "94"
```

```
SQL.UPDATE DB_Ptr, tbnam$, row$: Where$
```

**Sql.query** <cursor\_nvar>, <DB\_pointer\_nvar>, <table\_name\_sexp>, <columns\_sexp> {, <where\_sexp> {, <order\_sexp>}}

Queries a table of a previously-opened database for some specific data. The command returns a Cursor named <Cursor\_nvar> to be used in stepping through Query results. The <columns\_sexp> is a string expression with a list of the names of the columns to be returned. The column names must be separated by commas. An example is Columns\$ = "First\_name, Last\_name, Sex, Age". If you want to get the automatically incremented Row Index Column then include the "\_id" column name in your column list. Columns may be listed in any order. The column order used in the query will be the order in which the rows are returned.

The optional <where\_sexp> is an SQL expression string used to select which rows to return. In general, an SQL expression is of the form <Column Name> <operator> <Value>. For example, Where\$ = "First\_name = 'John' " Note that the Value must be contained in single quotes. Full details about the SQL expressions can be found [here](#). If the Where parameter is omitted, all rows will be returned.

The optional <order\_sexp> specifies the order in which the rows are to be returned. It identifies the column upon which the output rows are to be sorted. It also specifies whether the rows are to be sorted in ascending (ASC) or descending (DESC) order. For example, Order\$ = "Last\_Name ASC" would return the rows sorted by Last\_Name from A to Z. If the Order parameter is omitted, the rows are not sorted.

The result-sorting feature natively sorts upper and lower case separately, which is often not what is desired. However there is a feature that joins upper and lower case letters alphabetically in the sort. This can be invoked by replacing the "ASC" or "DESC" (ascending or descending) part of the <order\_sexp> argument with "COLLATE NOCASE ASC" OR "COLLATE NOCASE DESC" respectively. For example, Order\$ = "Last\_Name COLLATE NOCASE ASC" would return the rows sorted by Last\_Name from A to Z also if Last\_Name starts with lower case like "van der Meer".

COLLATE UNICODE sorts in Unicode order and COLLATE LOCALIZED in order of the current database language setting. If not specified by Sql.set\_locale the current system language setting is used.

If the Order parameter is present, the Where parameter must be present. If you want to return all rows, just set Where\$ = ""

**Sql.set\_locale** <DB\_pointer\_nvar>, <locale\_sexp>

The <locale\_sexp> specifies the output based on language and region. The locale specifies the language and region with standardized codes. The <locale\_sexp> is a string containing zero or more codes separated by underscores.

The function accepts up to three codes. The first must be a language code, such as "en", "de" or "ja".

The second must be a region or country code, such as "FR", "US", or "IN". Some language and country combinations can accept a third code, called the "variant code".

The function also accepts the standard three-letter codes and numeric codes for country or region. For example, "fr\_FR", "fr\_FRA", and "fr\_250" are all equivalent.

The locale takes only effect, if COLLATE LOCALIZED is part of the Sql.query <order\_sexp> statement. You may need to install your chosen language on the Android device first to get the full effect.

Example:

Sql.set\_locale DB\_Ptr, "de\_CH"

### Sql.exec <DB\_pointer\_nvar>, <command\_sexp>

Execute ANY non-query SQL command string ("CREATE TABLE", "DELETE", "INSERT", etc.) using a previously opened database.

Example:

```
dbname$ = "example.db"
Sql.open DB_Ptr, dbname$
tbname$ = "Birthdates"
Sql.drop_table DB_Ptr, tbname$ % Delete table if exists.
Sql.drop_table DB_Ptr, "SavedData" % Delete table if exists.
tbname$ = "Birthdates"
mColumns$ = "FirstName TEXT, Name TEXT "
CommandString$ = "CREATE TABLE IF NOT EXISTS " + tbname$ + "( " ~
+ "_id INTEGER PRIMARY KEY AUTOINCREMENT, "+ mColumns$ + " )"
Sql.exec DB_Ptr, CommandString$ % Create a table with two columns
CommandString$ = "ALTER TABLE "+tbname$+" ADD " + "Birthdate TEXT"
Sql.exec DB_Ptr, CommandString$ % Add a column
CommandString$ = "ALTER TABLE " + tbname$ + " RENAME TO " + "SavedData"
Sql.exec DB_Ptr, CommandString$ % Rename the table in "SavedData"
CommandString$ = "VACUUM" % Rebuild the database to smaller size.
Sql.exec DB_Ptr, CommandString$
Sql.insert DB_Ptr, "SavedData", "FirstName", "Mike", "Name", "Smith", "Birthdate", "2021-03-23"
Sql.insert DB_Ptr, "SavedData", "FirstName", "Lara", "Name", "Smith", "Birthdate", "2021-03-24"
↓
```

### Sql.raw\_query <cursor\_nvar>, <DB\_pointer\_nvar>, <query\_sexp>

Execute ANY SQL Query command using a previously opened database and return a Cursor for the results.

Example:

```
tbname$ = "SavedData"
sql$ = "SELECT sql FROM sqlite_master WHERE tbl_name = '"+tbname$+"' AND type = 'table';"
SQL.RAW_QUERY mCursor, DB_Ptr, sql$
SQL.NEXT xdone, mCursor, rawTableSql$
IF rawTableSql$ = ""
    PRINT "No database table with this name found!"
ELSE
    SPLIT.ALL a$, rawTableSql$, "[()]"
    SPLIT.ALL b$, a$[2], ","
    ARRAY.LENGTH al, b$[]
    DIM columns$(al-1), fTypes$(al-1)
    FOR i = 2 TO al
        SPLIT.ALL c$, b$[i], " "
        columns$(i-1) = c$[2]
        fTypes$(i-1) = c$[3]
    NEXT
    ENDIF
    index = 1
    JOIN.ALL columns$[], fields$, ","
    IF index THEN fields$ = "_id," + fields$
    SQL.QUERY cursor, DB_Ptr, tbname$, fields$
    delim$ = ";" % Different delimiter
    JOIN.ALL columns$[], fields$, delim$
    IF index THEN fields$ = " " + delim$ + fields$
```

```

csvLines$ = fields$ + CHR$(10)
xdone = 0
DO
  SQL.NEXT xdone, cursor, cv$[]
  !!b
  FOR n = 2 TO al % Add -1 at the end, if the index is not used.
    IF fTypes$(n-1) = "TEXT" THEN cv$(n) = CHR$(34) + cv$(n) + CHR$(34)
  NEXT
  !!e
  JOIN.ALL cv$[], fields$, delim$
  IF !xdone THEN csvLines$ += fields$ + CHR$(10)
UNTIL xdone
csvLines$ += CHR$(10)
csvLines$ = REPLACE$(csvLines$, CHR$(10) + CHR$(10), "") % Del. last LF
PRINT csvLines$

```

**SQL.PING** <result\_nvar>, <DB\_pointer\_nvar> {,<table\_name\_sexp> {,<column\_name\_sexp>}}

(A) SQL.PING r, db % ping only the database

(B) SQL.PING r, db, "mytable" % ping a table,

(C) SQL.PING r, db, "mytable", "col1" % ping a table and column

Returns the size (number of rows) of a database or table to r.

In case (A), this is the total number of user tables in the db.

The return code is as follows;

-1	The table name does not exist
-2	The table name exists but column does not exist in the table
≥0	The total number of tables in the database or rows in the table.

In case (C), this also means both table and column exists.

[ack:humpty0250]

## **SQL.CCL**

SQL.CCL clears the global SQL Cursor List.

## File Handling

**A new feature is the ability to cross the directory borders of BASIC! with the URI start "file://".**

Example:

```
FILE.ROOT dataPath$, "_Dcim"  
fn$ = "file://" + dataPath$
```

### About Android Directories

A distinction is made between **internal** and **external** directories.

**Internal** directories are owned by the App, their indicator is the App's package name (e.g. com.rfo.basic) in the absolute path. If the App is be deinstalled **all** the **internal** directories are deleted, too. The exceptions are **internal** directories upon **removable** SD cards before Android 4.4 KitKat (API19). This **BASIC!** version creates a **private internal** directory at first start.

**External** directories are member of the **public** part of Android's file system. From the view of the app it is controlled **externally**. So you need permissions finally in the file AndroidManifest.xml to access **external** directories. Please take a look at the compiler options, if you want to create an App.

The terms sd-ext or extSdcard are a case for a misunderstanding. These are expressions for **removable** SD cards.

**Removable** SD cards or USB sticks are normally **external** directories, too.

A bit odd in conjunction with this logic is the possibility to create an **internal** directory upon an **external** directory and sometimes also on a **removable** data carrier.

**Important: Internal directories on removable data carrier are public and automatically created by Android! Starting with Android 10 these are private.**

If you install the BASIC! Interpreter you get an **external public** directory rfo-basic.

If you deinstall BASIC! this directory and its contents is not removed. Only standard files like the code samples are changed after reinstalling.

If you create an APK with assets, read the article in the manual appendix about handling files from APKs and Resources (manual page 225).

Note, that you get not in all cases direct access to the files in Assets, because they are converted under certain circumstances. Maybe using sub directories is one of them.

It seems, that **RFO-BASIC 1.91** has the same issue. If you rename fly.gif in the data folder you should get access to the asset data folder, but no access!

In this case you have to copy these files to the different cache directory with Byte.open.

**If you need a dummy filename, use "/dev/null". Nothing is stored anywhere. But only available until Android 10.**

## File.root <full\_path\_svar>{, <dirType\_sexp>}

Returns the canonical path from the file system root to the default "<pref base drive>/rfo-basic/data", the default data directory, in <full\_path\_svar>. The <pref\_base\_drive> is expanded to the full absolute path from the file system root, "/".

The system constants in <dirType\_sexp> enables easy access to BASIC! and other folders:

\_Alarms, \_App, \_AppPath, \_Asset\_Cache, \_BasicSystem, \_Bluetooth, \_Cache, \_Data, \_Database, \_Dcim, \_Documents, \_Downloads, \_External, \_Internal, \_InternalOnExternal, \_InternalOnSdRemovable\*, \_Mnt, \_Movies, \_Music, \_Notifications, \_Pictures, \_Podcasts, \_ProgramPath, \_Ringtones, \_SdRemovable\*, \_Service, \_Source, \_SourceSamples, \_ScreenShots, \_Storage, \_System

\* Full available with Android 4.4 KitKat (API19) and later. At sooner APIs the interpreter searches for sdcard1, sdcard2, extSdcard and sd-ext.

### ToDo:

The system constants in <dirType\_sexp> enables easy access to BASIC! and other folders in conjunction to **Scoped Storage** (Android (10+) 11+):

\_AdocProvider, \_Alarms, \_App, \_AppPath, \_Asset\_Cache, \_BasicSystem, \_Bluetooth, \_Cache, \_Data, \_Database, \_Dcim, \_Documents, \_Downloads, \_External, \_FileProvider, \_Internal, \_InternalOnExternal, \_InternalOnSdRemovable\*, \_Mnt, \_Movies, \_Music, \_Notifications, \_Pictures, \_Podcasts, \_ProgramPath, \_Ringtones, \_SdRemovable\*, \_Service, \_Source, \_SourceSamples, \_ScreenShots, \_Storage, \_System

If an \_Internal... folder is not been created, it will create by the first call. The folder \_Documents is not created on some devices. With File.mkdir you can do it yourself. If you use the internal cache folder (\_Cache) this files will be ones that get deleted first, if the device runs low on storage. There is no indication when these files will be deleted. Note: you should not rely on the system deleting these files for you; you should always have a reasonable maximum, such as 1 MB, for the amount of space you consume with cache files, and prune those files when exceeding that space.

The \_Asset\_Cache folder as part of the \_Cache folder will be created at calling Byte.open myFileTable, "asset://" + myFileNamePath\$ if the folder does not exists.

Note: "file://" + <full\_path\_svar> is required if you want to use it directly in other BASIC! commands.

### Example:

```
FILE.ROOT dataPath$, "_Documents"
fn$ = "file://" + dataPath$
FILE.EXISTS ok, fn$
IF ok = 0
  FILE.ROOT newFolder$, "_External"
  newFolder$ = "file://" + newFolder$ + "/Documents"
  FILE.MKDIR newFolder$
ENDIF
```

**Experimental OliBasic3.00+:** RFO-BASIC and hBASIC ignore the slash at the beginning, because it is in these versions **not needed**.

If you want to use an absolute path in OliBasic your path\$ should begin with "file://" + ...

In the new OliBasic version a **slash** at the path beginning has the same function as "file://" as well as Android it does.

Thus file.root mPath\$ returns an absolute path, which can be used directly like this:  
mPath\$ = mPath\$ + "/" + "cartman.png". **If you get in trouble use "file://"** in case of absolute file or directory paths.

### **File.root.set.data <check\_svar>, <root\_path\_sexp>**

Sets the default data path to a different location by the parameter <root\_path\_sexp>. In every case relative paths of <root\_path\_sexp> are starting at (Base Directory)/rfo-basic/data. Absolute paths are also possible. The result of <check\_svar> returns the absolute path. If an error occurs the result of <check\_svar> begins with "Error:".

This command enables to structure the data of different programs.

It is also possible to store your data directly into the Adoc Provider or File Provider to share the program data with other apps.

Example:

```
! The program name is myTest.bas. You can create your default data path by
File.root.set.data ok$, "../projects/myTest"
PRINT ok$ % Returns ../rfo-basic/projects/myTest/data
```

### **File.root.set.databases <check\_svar>, <root\_path\_sexp>**

Sets the default database path to a different location by the parameter <root\_path\_sexp>. In every case relative paths of <root\_path\_sexp> are starting at (Base Directory)/rfo-basic/**data**. Absolute paths are also possible. The result of <check\_svar> returns the absolute path. If an error occurs the result of <check\_svar> begins with "Error:".

This command enables to structure the databases of different programs.

It is also possible to store your databases directly into the Adoc Provider or File Provider to share the program databases with other apps.

Example:

```
! The program name is myTest.bas. You can create your default database path by
File.root targetPath$, "_FileProvider"
File.root.set.databases ok$, targetPath$
PRINT ok$ % Returns /data/data/com.my.app/fileProvider/databases
BUNDLE.PUT bnd, "_FileP_Read", 1
BUNDLE.PUT bnd, "_FileP_Write", 1
PROVIDER bnd % Now the databases are accessible also from outside.
```

### **File.root.reset**

Resets the default data and database paths into the status at the program start.



### File.exists <lvar>, <path\_sexp>

Reports if the <path\_sexp> directory or file exists. If the directory or file does not exist, the <lvar> will contain zero. If the file or directory does exist, the <lvar> will be returned as non-zero. If the file or directory is readable <lvar> returns 2.0. If the file or directory is writeable <lvar> returns 3.0. If the file or directory is readable and writeable <lvar> returns 4.0.

The default path is "<pref base drive>/rfo-basic/data/".

It is also possible to use an URI with "file://" as start.

Sometimes you get a file path like "/external/images/media/556".

With "file://" + "/external/images/media/556", File.exists is able to handle it.

In this case <path\_sexp> returns a new absolute file path, **but only** if <path\_sexp> is a **single value** like fn\$ and **not** a simple string expression.

File.exists has no access to **assets** and **resources** in conjunction with APKs.

### File.md5 <svar>, <path\_sexp>

Returns the MD5 hash of the file specified by <path\_sexp>.

It is only useful, if a compare of the contents of a file is needed, because this algorithm is not secure but faster.

### File.sha <svar>, <path\_sexp>{, <algorithm\_sexp>}

Returns the SHA hash of the file specified by <path\_sexp> and the algorithm <algorithm\_sexp>. Possible are \_SHA-1, \_SHA-224, \_SHA-256, \_SHA-384, \_SHA-512, \_SHA-512/224 or \_SHA-512/256. Default is \_SHA-256.

Secure hash algorithms – SHA-1(insecure!), SHA-224, SHA-256, SHA-384, SHA-512 - for computing a condensed representation of electronic data (message). When a message of any length less than  $2^{64}$  bits (for SHA-1, SHA-224, and SHA-256) or less than  $2^{128}$  (for SHA-384 and SHA-512) is input to a hash algorithm, the result is an output called a message digest. A message digest ranges in length from 160 to 512 bits, depending on the algorithm.

## **File.copy <sourcePath\_sexp>, <targetPath\_sexp>{, <modes\_sexp>}**

Copies a File or Directory

On the default mode the copy fails if the target file exists. In this case use File.exists before.

Directories can be copied. However, files inside the directory are not copied, so the new directory is empty even when the original directory contains files.

When copying a symbolic link, the target of the link is copied. If you want to copy the link itself, and not the contents of the link, specify either the `_ReplaceExisting` option.

The following Options are additional supported:

`_ReplaceExisting` – Performs the copy even when the target file already exists. If the target is a symbolic link, the link itself is copied (and not the target of the link). If the target is a non-empty directory, the copy fails with the `FileAlreadyExistsException` error.

`_CopyAttributes` – Copies the file attributes associated with the file to the target file. The exact file attributes supported are file system and platform dependent, but last-modified-time is supported across platforms and is copied to the target file.

To specify one or two options put a comma separated string into `<modes_sexp>` like this `"_ReplaceExisting, _CopyAttributes"`. The delimiter is the comma `,`.

This command is only supported beginning with Android 8+ (API 26+).

See also `File.exists`

## **File.move <sourcePath\_sexp>, <targetPath\_sexp>{, <modes\_sexp>}**

Moves a File or Directory

On the default mode the move fails if the target file exists. In this case use File.exists before.

Empty directories can be moved. If the directory is not empty, the move is allowed when the directory can be moved without moving the contents of that directory. On the Android system, moving a directory within the same partition generally consists of renaming the directory. In that situation, this method works even when the directory contains files.

The following Options are additional supported:

\_ReplaceExisting – Performs the move even when the target file already exists. If the target is a symbolic link, the symbolic link is replaced but what it points to is not affected.

\_AtomicMove – Performs the move as an atomic file operation. If the file system does not support an atomic move, an exception is thrown. With an \_AtomicMove you can move a file into a directory and be guaranteed that any process watching the directory accesses a complete file.

To specify one or two options put a comma separated string into <modes\_sexp> like this "\_ReplaceExisting, \_AtomicMove". The delimiter is the comma ",". **funktioniert nicht**

This command is only supported beginning with Android 8+ (API 26+).

See also File.exists

## **File.replace <startPath\_sexp>, <replace\_list\_nexp>**

Replaces a String in a file or files within directories.

This will start at a location, and if that location is a file it will process it. If the location is a directory it will process all of the files in the directory. If the directory contains other directories, it will continue to recursively process until all files in all sub directories of the original location have been processed.

The entries of the String List with its pointer <replace\_list\_nexp> have to be in order of replace type, the string to be replaced and the replacement. This order can be used multiple times to replace more than one text string. It is replaced in turn.

Possible replace types are:

### **\_Replace**

Replaces each substring of this string that matches the literal target sequence with the specified literal replacement sequence. There placement proceeds from the beginning of the string to the end, for example, replacing "aa" with "b" in the string "aaa" will result in "ba" rather than "ab".

### **\_ReplaceIgnoreCase**

Replaces each substring of this string that matches the literal target sequence with the specified literal replacement sequence. There placement proceeds from the beginning of the string to the end, for example, replacing "aa" with "b" in the string "Aaa" will result in "ba" rather than "ab".

### **\_ReplaceAll**

Replaces each substring of this string that matches the given regular expression with the given replacement.

### **\_ReplaceFirst**

Replaces the first substring of this string that matches the given regular expression with the given replacement.

See also `Replace$()`, `List.match`

Example:

```
LIST.CREATE s, rpl
LIST.ADD rpl, "_ReplaceFirst", "m.c", "g.c", "_Replace", "good", "bad"
FILE.REPLACE "testDir", rpl
```

### **File.lastmodified <nvar>, <path\_sexp>**

Returns the time when this file was last modified, measured in milliseconds since January 1st, 1970, midnight GMT.

If something went wrong <nvar> returns 1, maybe you try to get the time from a Resource or Asset file.

Example:

```
FILE.ROOT pp$ , "_SourceSamples"
FILE.LASTMODIFIED lm, "file://" + pp$ + "/" + "f01_commands.bas"
? USING$("", "%TF %tT", lm, lm) % Returns "2019-06-30 21:01:01"
FILE.SET.LASTMODIFIED lm, "file://" + pp$ + "/" + "f01_commands.bas", lm + 6000
? USING$("", "%TF %tT", lm, lm) % Returns "2019-06-30 21:01:07"
```

### **File.set.lastmodified <nvar>, <path\_sexp>, <new\_time\_nexp>**

The parameter <new\_time\_nexp> sets the new time when this file should be last modified. The time is measured in milliseconds since January 1st, 1970, midnight GMT.

As a feedback <nvar> returns the new measured time.

If something went wrong <nvar> returns 1, maybe you try to get the time from a Resource or Asset file.

### **File.absolute <absolute\_svar>, <path\_sexp>**

Returns the absolute file path by <absolute\_svar>.

It tries to convert a document path beginning with "content://" into an absolute file path also. In this case min. KitKat 4.4 (API 19) is needed. If is ":open uri with ADOC.Read or ADOC.Write" returned, use these ADOC commands for access.

**File.dir** <path\_sexp>, Array\$[] {{{{,<dirmark\_sexp>,<timeStamp\_nexp>},<recursive\_nexp>}, <type\_sexp>}

Returns the names of the files and directories in the path specified by <path\_sexp>. The path is relative to "<pref base drive>/rfo-basic/data/".

When you initially add "**asset://**", you will get the directories and files from the **APK** assets. Depending on the structure of the assets, the distinction between directory or file is only possible via the point. There is no opportunity to get file time stamps in this case.

Keep also in mind, that there are **no empty directories** in assets!

The names are placed into Array\$[]. The array is sorted alphabetically with the directories at the top of the list. If the array exists, it is overwritten, otherwise a new array is created. The result is always a one-dimensional array.

A directory is identified by a marker appended to its name. The default marker is the string "(d)". You can change the marker with the optional directory mark parameter <dirmark\_sexp>. If you do not want directories to be marked, set <dirmark\_sexp> to an empty string, "".

**Dir** is a valid alias for this command.

If the directory is empty, File.dir returns an array with one item and a string with one space (" ") in it.

Options of <timeStamp\_nexp>:

- 0 no time stamp (default)
- 1 with time stamp as time in milliseconds + ":" + file name, but unsorted
- 2 with time stamp as time in milliseconds + ":" + file name, sorted in ascending order
- 3 with time stamp as time in milliseconds + ":" + file name, sorted in descending order

If <recursive\_nexp> is > 0 all sub directories are searched as well. Default is 0.

If only files or directories are needed use <type\_sexp> with "\_F" for files and "\_D" for directories. Default is "\_DF".

Example:

```
FILE.ROOT path$, "_Mnt"  
FILE.DIR path$, dirArray$[], "", 0 , 1 , "_F"  
LIST.CREATE s, dirltems  
LIST.ADD.ARRAY dirltems, dirArray$[]  
LIST.CREATE s, dirFilteredItems  
LIST.MATCH , dirFilteredItems, dirltems, "png",,,, "_Ends_With_IgnoreCase"  
DEBUG.ON  
DEBUG.DUMP.LIST dirFilteredItems
```

**File.delete** <lvar>, <path\_sexp>{,<recursive\_nexp>}

The file or directory at <path\_sexp> will be deleted, if it exists. If the file or directory did not exist before the Delete, the <lvar> will contain zero. If the file or directory did exist and was deleted, the <lvar> will be returned as not zero.

The default path is "<pref base drive>/rfo-basic/data/".

If <recursive\_nexp> is 1 all sub directories and files are deleted as well.

If <recursive\_nexp> is 2 the directory specified by <path\_sexp> is deleted as well.

Default is 0.

**File.select** <filePath\_svar>, <startPath\_sexp>{, <settings\_bundle\_nexp>}

Returns a chosen directory or file path by <filePath\_svar>. The selection process begins at <startPath\_sexp>.

If you use buttons, the result is different. See the table below.

The optional layout bundle <layout\_bundle\_nexp> controls the File.select output layout:

Table of setting options		
Key	Value	Description
<b>_Style</b>	_Default	Default theme
	_Dark	Dark theme and Autosize mode
	_Bright	Bright theme and Autosize mode
<b>_Icon</b>	String	File path of a header icon
<b>_Action</b>	String	Action as start of the title.
<b>_PositionH</b>	_Left	
	_Center	Default
	_Right	
<b>_PositionV</b>	_Top	
	_Center	Default
	_Bottom	
<b>_DisplayPath</b>	numeric	If it is 0 no absolute path will be displayed in the title. If > 0 the absolute path will be displayed. (Default)
<b>_EndsWith</b>	String like "?/??/?"	If the string is not empty, only files with given endings will be returned. A string like "jpg/gif/mp3 ..." with endings like ". jpg", ". gif", ".mp3" ... will only return files with theses endings. The delimiter is the slash "/", because the comma can be part of a file name. If the string is "-1" only directories will be returned.
<b>_UpperStart</b>	numeric	If it is 0 no path upper the start path will be displayed.(Default) If > 0 paths upper the start path will be displayed.
<b>_OnlyDirs</b>	numeric	If it is > 0 only directories are displayed. The default is 0.
<b>_Button1</b>	Text of Button 1	<b>Positive</b> button Default is "⬅ ➡". Pressing returns the next upper directory if possible.
<b>_Button1Size</b>	numeric	

Table of setting options		
Key	Value	Description
<b>_Button1Color</b>	{Alpha,}Red,Green,Blue (comma delimited string) or _{Alpha,}ColorName ({comma delim.} string) or #{hn}hnhnhn (hex. string)	
<b>_Button2</b>	Name of Button 2	<b>Neutral</b> button Default is "". You could use "+" + CHR\$(128194) (  ) as an example. Pressing returns a "+" and the current absolute path follows.
<b>_Button2Size</b>	numeric	
<b>_Button2Color</b>	{Alpha,}Red,Green,Blue (comma delimited string) or _{Alpha,}ColorName ({comma delim.} string) or #{hn}hnhnhn (hex. string)	
<b>_Button3</b>	Name of Button 3	<b>Negative</b> button Default is "X". Pressing returns a "-" and the current absolute path follows.
<b>_Button3Size</b>	numeric	
<b>_Button3Color</b>	{Alpha,}Red,Green,Blue (comma delimited string) or _{Alpha,}ColorName ({comma delim.} string) or #{hn}hnhnhn (hex. string)	
<b>_Cancelable</b>	numeric	If > 0 the dialog is cancel-able. Default is 1.



## Byte.open {r|w|a}, <file\_table\_nvar>, <path\_sexp>

The file specified by the path string expression <path\_sexp> is opened. If the path is a URL starting with "http..." then an Internet file is opened. Otherwise, the <path\_sexp> string is appended to the default path "<pref base drive>/rfo-basic/data/".

If the URL starts with "asset://" + myFileNamePath\$ the \_Asset\_Cache folder as part of the \_Cache folder will be created if the folder does not exist. So a later Byte.copy or some other commands are able to access the file from the App's Asset Cache Directory. Larger files should be deleted after use.

The first parameter is a single character that sets the I/O mode for this file:

Parameter	Mode	Notes
r	read	File exists: Reads from the start of the file. File does not exist: Error (see below).
w	write	File exists: Writes from the start of the file. Writes over any existing data. File does not exist: Creates a new file. Writes from the start of the file.
a	append	File exists: Writing starts after the last line in the file. File does not exist: Creates a new file. Writes from the start of the file.

A file table number is placed into the numeric variable <file\_table\_nvar>. This value is for use in subsequent **Byte.read.\***, **Byte.write.\***, **Byte.eof**, **Byte.position.\***, **Byte.truncate**, **Byte.copy**, or **Byte.close** commands.

If a file being opened for read does not exist then the <file\_table\_nvar> will be set to -1. The BASIC! program can check for this and either create the file or report the error to the user. Information about the error is available from the **GETERROR\$()** function.

GitHub#249

**Byte.copy** <file\_table\_nexp>, <output\_file\_sexp>{,<append\_nexp>},  
<break\_nexp>}

Copies the previously open input file represented by <file\_table\_nexp> to the file whose path is specified by <output\_file\_sexp>. The default path is "<pref base drive>/rfo-basic/data/".

If <file\_table\_nexp> = -1 then a run-time error will be thrown.

All bytes from the current position of the input file to its end are copied to the to the output file. Both files are then closed.

If you have read from the input file, and you want to copy the whole file, you must reset the file position to 0 with **Byte.position.set**. However, if you have changed the file mark with **Byte.position.mark**, or if you reading a non-local (internet) file, you can't reset the file position to 0. Instead, you must close and reopen the file.

You should use **Byte.copy** if you are using Byte I/O for the sole purpose of copying. It is thousands (literally) of times faster than using **Byte.read/Byte.write**.

If you want to append the input file content to the output file, the optional <append\_nexp> parameter has to be > 0. Default is 0.

If the output file does not exists in both cases, a new file will be created before writing.

If <break\_nexp> is greater than zero, Byte.copy can be interrupted by an interrupt.

**Byte.read.buffer** <file\_table\_nexp>, <count\_nexp>, <buffer\_svar>{,  
<charset\_sexp>}

Reads the specified number of bytes (<count\_nexp>) into the buffer string variable (<buffer\_svar>) from the file. The string length (len(<buffer\_svar>)) will be the number of bytes actually read. If the end of file is reached, the string length may be less than the requested count.

A buffer string is a special use of the BASIC! string. Each character of a string is 16 bits. When used as a buffer, one byte of data is written into the lower 8 bits of each 16-bit character. The upper 8 bits are 0. Extract the binary data from the string, one byte at a time, with the **ASCII()** or **UCODE()** functions.

The format of the buffer string read by this command is compatible with the **DECODE\$()** function. If you know that part of your data contains an encoded string, you can extract the substring (using a function like **MID\$()**), then pass the substring to **DECODE\$()** to convert it to a BASIC! String.

Using the optional <charset\_sexp> you can choose between following character sets: "\_US-ASCII" and "\_ISO-8859-1"(default).

In most cases the command GrabFile is the better choice.

## **Byte.write.buffer** <file\_table\_nexp>, <buffer\_sexp>{, <charset\_sexp>}

Writes the entire contents of the string expression to the file. The string is assumed to be a buffer string holding binary data, as described in **Byte.read.buffer**. The writer discards the upper 8 bits of each 16-bit character, writing one byte to the file for each character in the string.

The **Byte.read.buffer** command and the **ENCODE\$()** function always create these "buffer strings". You can construct one by using, for example, the **CHR\$()** function with values less than 256.

If you use only ASCII characters in a string, you can use this function to write the string to a file. The output is the same as if you had written it with **Text.writeIn**, except that it will have no added newline.

Using the optional <charset\_sexp> you can choose between following character sets:

"\_US-ASCII", "\_UTF-8", "\_UTF-16", "\_UTF-16BE", "\_UTF-16LE" and "\_ISO-8859-1"(default).

If you want to save international text use "\_UTF-8".

This command expects only character codes up to 255 for "\_US-ASCII" or for binary data "\_ISO-8859-1" and writes only one byte per character.

If you want to convert an integer to a byte array use bit shifting like this:

```
FN.DEF integer2Str$( val)
    b0$=CHR$(band(val,255))
    b1$=CHR$(band(shift(val,8),255))
    b2$=CHR$(band(shift(val,16),255))
    b3$=CHR$(band(shift(val,24),255))
    FN.RTN b0$+b1$+b2$+b3$
FN.END
```

Using "\_UTF-8", "\_UTF-16", "\_UTF-16BE", "\_UTF-16LE" with **Byte.write.buffer** write byte arrays with up to 4 bytes for each character.

If you get in trouble with binary data using **Byte.write.buffer** try **Byte.write.byte** with your string. It is slow but it interprets character codes over 255 differently.

See also **Byte.write.byte**, **GrabFile**, **GrabURL**

### **GrabFile** <result\_svar>, <path\_sexp>{, <unicode\_flag\_lexp>|<charset\_sexp>}

Copies the entire contents of the file at <path\_sexp> to the string variable <result\_svar>. By default, **GrabFile** assumes that the file contains binary bytes or ASCII characters. If the optional <unicode\_flag\_lexp> evaluates to true (a non-zero numeric value), **GrabFile** can read Unicode text.

If you change <unicode\_flag\_lexp> to <charset\_sexp> you can choose between following character sets: "\_US-ASCII"(1), "\_UTF-8"(1), "\_UTF-16", "\_UTF-16BE", "\_UTF-16LE" and "\_ISO-8859-1"(0).

If the file does not exist or cannot be opened, the <result\_svar> is set to the empty string, "", and you can use the **GETERROR\$()** function to get more information. If the file is empty, the <result\_svar> is an empty string, "", but **GETERROR\$()** returns "No error". For text files, either ASCII or Unicode, the **Split** command can be used to split the <result\_svar> into an array of lines. **GrabFile** can also be used grab the contents of a text file for direct use with **Text.input**:

```
GRABFILE text$, "MyJournal.txt"  
TEXT.INPUT EditedText$, text$
```

### **GrabURI** <result\_svar>, <uri\_sexp>{, <unicode\_flag\_lexp>|<charset\_sexp>}

Copies the entire contents of the file at <path\_sexp> to the string variable <result\_svar>. By default, **GrabURI** assumes that the file contains binary bytes or ASCII characters. If the optional <unicode\_flag\_lexp> evaluates to true (a non-zero numeric value), **GrabURI** can read Unicode text.

If you change <unicode\_flag\_lexp> to <charset\_sexp> you can choose between following character sets: "\_US-ASCII"(1), "\_UTF-8"(1), "\_UTF-16", "\_UTF-16BE", "\_UTF-16LE" and "\_ISO-8859-1"(0).

If the file does not exist or cannot be opened, the <result\_svar> is set to the empty string, "", and you can use the **GETERROR\$()** function to get more information. If the file is empty, the <result\_svar> is an empty string, "", but **GETERROR\$()** returns "No error". For text files, either ASCII or Unicode, the **Split** command can be used to split the <result\_svar> into an array of lines. **GrabFile** can also be used grab the contents of a text file for direct use with **Text.input**:

```
GRABFILE text$, "MyJournal.txt"  
TEXT.INPUT EditedText$, text$
```

**GrabURL** <result\_svar>, <url\_sexp>{{, <timeout\_nexp>},<unicode\_flag\_lexp>|<charset\_sexp>}

Copies the entire source text of the URL <url\_sexp> to the string variable <result\_svar>. The URL may specify an Internet resource or a local file. **Cached files will be ignored.** If the URL does not exist or the data cannot be read, the <result\_svar> is set to the empty string, "", and you can use the **GETERROR\$()** function to get more information.

If the optional <timeout\_nexp> parameter is non-zero, it specifies a time-out in milliseconds. **Thus 0 specified no time-out.** This is meaningful only if the URL names a resource on a remote host. If the time-out time elapses and host does not connect or does not return any data, **GETERROR\$** reports a socket timeout. **By default, GrabURL assumes that the file contains Unicode text.** If the optional <unicode\_flag\_lexp> evaluates to false (a zero numeric value), **GrabURL** can read binary bytes or ASCII characters.

**Attention: The default <unicode\_flag\_lexp> setting is the opposite of GrabFile!**

If you change <unicode\_flag\_lexp> to <charset\_sexp> you can choose between following character sets: "\_US-ASCII"(1), "\_UTF-8"(1), "\_UTF-16", "\_UTF-16BE", "\_UTF-16LE" and "\_ISO-8859-1"(0).

If the named resource is empty, the <result\_svar> is empty, "", and **GETERROR\$()** returns "No error".

The "Split" command can be used to split the <result\_svar> into an array of lines for ASCII or Unicode text.

## FTP Client

These FTP commands implement a FTP Client

**Ftp.open <url\_sexp>, <port\_nexp>, <user\_sexp>, <pw\_sexp>{, <ok\_svar>}**

Connects to the specified url and port. Logs onto the server using the specified user name and password.

The optional <ok\_svar> returns an empty string if nothing fails. Otherwise the error message is returned.

For example:

```
ftp.open "ftp.dlptest.com", 21, "dlpuser", "rNrKYTX9g7z3RgJRmxWuGHbeu"
```

see <https://dlptest.com/ftp-test/>

You can also write:

```
ftp.open "ftp://ftp.dlptest.com", 21, "dlpuser", "rNrKYTX9g7z3RgJRmxWuGHbeu"
```

Or:

```
ftp.open "ftps://myServer.net", 2221, "demo", "demo"
```

```
ftp.open "ftpes://192.168.1.2", 2221, "demo", "demo"
```

In the last two cases you open a ftp over Transport Layer Security (TLS) connection, so called FTPS. This implementation works in explicit mode (also known as FTPES).

Working as a client with implicit mode is not supported by Android.

For a FTP/FTPES/FTPS server on Android is the **Wifi FTP Server** recommended.

(id=com.medhaapps.wififtpserver or id=com.medhaapps.wififtpserver.pro)

If you want start and stop a FTP server, you can use the app **FTP-Server**, also.

(id=com.theolivetree.ftpserver or id=com.theolivetree.ftpserverpro)

Start and Stop by the following intents:

```
com.theolivetree.ftpserver.StartFtpServer or  
com.theolivetree.ftpserver.StartFtpServerPro  
com.theolivetree.ftpserver.StopFtpServer or  
com.theolivetree.ftpserver.StopFtpServerPro
```

If you want to use SFTP use the App **andFTP** (id=lysesoft.andftp and

id=lysesoft.andftp[Key only]). This App can be controlled by intents.

More informations at <http://www.lysesoft.com/products/andftp/index.html>.

For a SFTP server on Android is the **Ssh Server** recommended.

(id=com.theolivetree.sshserver or id=com.theolivetree.sshserverpro)

Start and Stop by the following intents:

```
com.theolivetree.sshserver.StartSshServer or com.theolivetree.sshserverpro.StartSshServer  
com.theolivetree.sshserver.StopSshServer or com.theolivetree.sshserverpro.StopSshServer
```

**Ftp.close{, <ok\_svar>}**

Disconnects from the FTP server.

The optional <ok\_svar> returns an empty string if nothing fails. Otherwise the error message is returned.

**Ftp.put <source\_sexp>, <destination\_sexp>{, <ok\_svar>}**

Uploads specified source file to the specified destination file on connected ftp server.

The source file is relative to the directory, "<pref base drive>/rfo-basic/data/" If you want to upload a BASIC! source file, the file name string would be: "../source/xxxx.bas".

The destination file is relative to the current working directory on the server. If you want to upload to a sub directory of the current working directory, specify the path to that directory. For example, if there is a sub directory named "etc" then the filename, "/etc/name" would upload the file into that sub directory.

The optional <ok\_svar> returns an empty string if nothing fails. Otherwise the error message is returned.

### **Ftp.get <source\_sexp>, <destination\_sexp>{, <ok\_svar>}**

The source file on the connected ftp server is downloaded to the specified destination file on the Android device.

You can specify a subdirectory in the server source file string.

The destination file path is relative to "<pref base drive>/rfo-basic/data/" If you want to download a BASIC! source file, the path would be, "../source/xxx.bas".

The optional <ok\_svar> returns an empty string if nothing fails. Otherwise the error message is returned.

### **Ftp.dir <list\_nvar> {{{,<dirmark\_sexp>},<timeStamp\_nexp>}, <ok\_svar>}**

Creates a list of the names of the files and directories in the current working directory and places it in a BASIC! List data structure. A pointer to the new List is returned in the variable <list\_nvar>.

A directory is identified by a marker appended to its name. The default marker is the string "(d)". You can change the marker with the optional directory mark parameter <dirmark\_sexp>. If you do not want directories to be marked, set <dirmark\_sexp> to an empty string, "".

Options of <timeStamp\_nexp>:

- 0 no time stamp (default)
- 1 with time stamp as time in milliseconds + ":" + file name, but unsorted
- 2 with time stamp as time in milliseconds + ":" + file name, sorted in ascending order
- 3 with time stamp as time in milliseconds + ":" + file name, sorted in descending order

The optional <ok\_svar> returns an empty string if nothing fails. Otherwise the error message is returned.

The following code can be used to print the file names in that list:

```
ftp.dir file_list
list.size file_list,size

for i = 1 to size
    list.get file_list,i,name$
    print name$
next i
```

GitHub#244

### **Ftp.cd <new\_directory\_sexp>{, <ok\_svar>}**

Changes the current working directory to the specified new directory.

The optional <ok\_svar> returns an empty string if nothing fails. Otherwise the error message is returned.

### **Ftp.rename <old\_filename\_sexp>, <new\_filename\_sexp>{, <ok\_svar>}**

Renames the specified old filename to the specified new file name.

The optional <ok\_svar> returns an empty string if nothing fails. Otherwise the error message is returned.

### **Ftp.delete <filename\_sexp>{, <ok\_svar>}**

Deletes the specified file.

The optional <ok\_svar> returns an empty string if nothing fails. Otherwise the error message is returned.

### **Ftp.rmdir <directory\_sexp>{, <ok\_svar>}**

Removes (deletes) the specified directory if and only if that directory is empty.

The optional <ok\_svar> returns an empty string if nothing fails. Otherwise the error message is returned.

### **Ftp.mkdir <directory\_sexp>{, <ok\_svar>}**

Creates a new directory of the specified name.

The optional <ok\_svar> returns an empty string if nothing fails. Otherwise the error message is returned.



## FTP Server

These commands control the built-in ftp server.

This server does **not** support a ftp over Transport Layer Security (TLS) connection, so called FTPS until now. Because of this, these commands differ in the name of Hbasic commands.

FTP.server.set Configures the server. (must restart server after)

FTP.server.start Starts the server.

FTP.server.stop Stops the server.

All commands return an exit code (not optional) which will be one of the following;

Return Codes	
0	Success
1	Could not start server
2	Server already running
3	Could not stop server
4	Server already closed
5	Bad port number (port number must be 1025 to 65535)
6	No internet permission

### FTP.server.set return\_code\_nvar{{{port\_nexp}, username\_sexp}, password\_sexp}, welcome\_string\_sexp}

Configures the server. The return code will be delivered by return\_code\_nvar. To set the servers control port between 1025 and 65535 use port\_nexp. The default port is 2121. The login user name from client to server is specified by the optional username\_sexp. Default name is "olibasic". The login password from client to server is specified by the optional password\_sexp. Default password is "olibasic".

Note that some ftp-clients do not allow to have empty usernames and/or passwords.

Use lowercase for all parameters.

The optional welcome string from server to client can be set by welcome\_string\_sexp. Do not put any newlines ("\n") inside the welcome string.

The default message is: "Welcome to the OliBasic's FTP-Server".

**The configuration does not take place until a new server starts.** So do this before starting the server.

Example:

```
FTP.server.set rc, "john", "mypass"  
FTP.server.set rc, "john", p$, "Welcome to myApp's Server"  
FTP.server.set ,newpassword$
```

### **FTP.server.start return\_code\_nvar{, ip\_svar}**

Starts the server. The return code will be delivered by return\_code\_nvar. The optional ip\_svar returns the server's ip address plus the port number used as "d.d.d.d:port" like 192.168.1.105:2121.

Example:

```
FTP.server.start rc, IP$  
IF rc > 0 THEN PRINT "Error" ELSE PRINT "Server started on IP$"
```

### **FTP.server.stop return\_code\_nvar**

Stops the server. The return code will be delivered by return\_code\_nvar.

Example:

```
FTP.server.stop rc  
IF rc <> 0 THEN PRINT "Error" ELSE PRINT "Server stopped"
```

## Documents

Android supports **documents**. An Android document can be a directory, a file, a database or something else. Each document contains content, that is provided by a content provider. A path pointing at a content begins with "content://".

A content provider gives access to the own or data to other applications also. It can as an example be a file system or a cloud storage like Google Drive.

OliBasic does support an Adoc Provider and a File Provider provider today, but you can provide also data in a file saved into an **external** directory or a supported cloud storage. **External** means public file access. SD cards and USB sticks are **removable** data carrier.

Each valid document path gives you access to the document name and the default size, but not in all cases to the data.

Handling documents is a little different in opposite to files.

For better understanding we will look at a cloud storage first.

A cloud storage is connected by a network.

If you use a mobile network, the link can be broken. To prevent data loss it needs an exact status each time.

If you want a directory document, you have to use Adoc.save with a document type named "\_Dir".

A new file needs also Adoc.save.

In this case you can preset the document type and its name.

Creating a new directory in its workflow is possible too.

Now the document has to be filled by Adoc.write.

This command uses strings.

If you want to copy or save binary data use "\_ISO-8859-1" as character set.

It is time to try to read the stored content.

We start Adoc.open to choose a document.

If the network connection is not broken, we get a document path beginning with "content://".

Adoc.read gives the possibility to get the content as a string.

If you want to copy a document read and put the content into a string with the "\_ISO-8859-1" character set and write the content into a new document created by the Adoc.save dialog with the same character set. Try to enumerate the existing of the created document.

If a document has to be renamed, use Adoc.rename or its workaround.

A content path is different to a file path.

Sometimes it looks like a readable file path, but in cases of cloud storage, last documents, downloads etc. it is coded or/and encrypted.

File.absolute try to convert the document path into an absolute file path.

If it fails using a valid document path, returns only the document name without any slash ("/").

Selecting a document in the Downloads document directory returns a file path, if the document name equals with a first level file name.

If it returns a valid absolute file path, you can operate with normal file dependent commands.

Otherwise copy documents into files to handle with.

In case of Adoc (Documents) providers the permissions have to be granted. In some commands there is a <grant\_perm\_nexp> parameter. Default is 1. So permissions will be granted by default. In case of using File Providers the <grant\_perm\_nexp> parameter should be set to 0. To control the access to your own providers see also the command PROVIDER.

What is the outlook?

Google will take more care about user's security.

The user will be more involved to confirm actions with possible data risks.

That is not what developers normally want.

They like paths, which can be extend by easy readable directory and file names.

The consequence is storing selected document paths permanently in a private file folder created by File.root path\$, "\_Internal". However, there is no guarantee that the document path will be the same after a restart or an extended period of time between runs of the application.

**Known issues:**

**<startPath\_sexp> does not work properly in all cases.**

If you have a removable device like a SD-card "/storage/9016-4EF9" take look that the device's content path ends with "%3A" like

"content://com.android.externalstorage.documents/document/9016-4EF9%3A".

**Adoc.path <documentPath\_svar>, <filePath\_sexp>**

Returns a document path beginning with "content://" by the given file path.

If an error occurs or it is not possible an empty string ("" ) is returned.

**Adoc.open <documentPath\_svar>|Array\$[]{{, <startPath\_sexp>}, <mimeType\_sexp>}**

Opens a system dialog for opening a document and returns the selected path by <documentPath\_svar>. If <documentPath\_svar> returns an empty string (""), the operation was not successful. If an Array\$[] is used multiselection is possible. If the selection in this case is failed, the first array item returns an empty string ("").

An existing start path will be defined by <startPath\_sexp>. An empty string let the document browser start at the default data path. If this attribute is not used the start begins at the last visited location.

Android stores each document into a database and is also interested in the document type. To shrink the number of select-able documents use <mimeType\_sexp> to specify the wanted types.

It grants also permissions to the selected document until the system will be restarted.

**So restart your system and test your app again before publishing.**

The default MIME type is "\*\*/\*".

Common types are "image/png", "image/jpg", "text/plain" etc..

If a new document directory is needed, use "\_Dir". Unfortunately ADOC.OPEN **is not able** to select a **directory**.

Min. KitKat 4.4 (API 19) is needed.

Example:

```
! As a file browser
startPath$ = "" % Default data path
ADOC.OPEN documentName$, startPath$
FILE.ABSOLUTE fileName$, documentName$
```

```

GRABFILE result$, fileName$
PRINT result$

or
! As a document browser
startPath$ = "" % Default data path
ADOC.OPEN documentName$, startPath$
ADOC.READ result$, documentName$
PRINT result$

or
! As a document browser perhaps on Google Drive
FILE.ROOT path$, "_Internal"
absoluteDocsPath$ = "file://" + path$ + "docs.bn"
FILE.EXISTS ok, absoluteDocsPath$
IF ok
    BUNDLE.LOAD mDocs, absoluteDocsPath$
    ! bundle.clear mDocs % If something went wrong at the first tries
    ! In this case do not forget to delete all created Documents also
    ! Document Directories by the Google Drive App before.
    BUNDLE.CONTAIN mDocs, "myDocDirectoryOnGoogleDrive", ex
    IF ex THEN BUNDLE.GET mDocs, "myDocDirectoryOnGoogleDrive", startPath$
ENDIF
IF !ok | !ex
    rootDir$ = "myDocDirectory"
    ADOC.SAVE startPath$, rootDir$, "", "_Dir"
    PRINT "startPath$", startPath$
    BUNDLE.PUT mDocs, "myDocDirectoryOnGoogleDrive", startPath$
    BUNDLE.SAVE mDocs, absoluteDocsPath$
ENDIF
newDocName$ = "myFirstDocument.txt"
BUNDLE.CONTAIN mDocs, "." + newDocName$, ex
IF ex
    message$ = "Take care, because\n you are able to\n " ~
    + "create a second one\n with the same name"
    DIALOG.MESSAGE "Document Exists", message$, sel , "SKIP", "OK"
ENDIF
IF sel <> 1
    ADOC.SAVE documentPath$, newDocName$, startPath$, "text/plain"
    BUNDLE.PUT mDocs, "." + newDocName$, documentPath$
    BUNDLE.SAVE mDocs, absoluteDocsPath$
ENDIF
ADOC.EXISTS ok, documentPath$
IF ok THEN ADOC.WRITE ws, documentPath$, "My First Text On Google Drive"
ADOC.OPEN documentPath$ ,startPath$
ADOC.READ ok, result$, documentPath$ %, "_UTF-8"
PRINT result$, ok
END
! Before the third run change your root Document Directory name
! by the Google Drive App.
! You should see, that your Document is still selectable.
! With Adoc.name you can update your Docs bundle.

```

**Adoc.save** <documentPath\_svar>, <documentName\_svar>{{, <startPath\_sexp>}, <mimeType\_sexp>}

Opens a system dialog for saving a document and returns the created or selected path by <documentPath\_svar>. If <documentPath\_svar> returns an empty string (""), the operation was not successful.

<documentName\_svar> presents the new document name.

An existing start path will be defined by <startPath\_sexp>. An empty string let the document browser start at the default data path. If this attribute is not used the start begins at the last visited location.

Android tries to put each document into a database and is also interested in the document type. To shrink the number of select-able documents use <mimeType\_sexp> to specify the wanted types.

The default MIME type is "\*/\*".

Common types are "image/png", "image/jpg", "text/plain" etc..

If a new document directory is needed, use "\_Dir".

In newer Android versions a menu item to create a new directory is added in the document browser.

Min. KitKat 4.4 (API 19) is needed.

**Adoc.get** <documentPath\_svar>|Array\$[]{{, <startPath\_sexp>}, <mimeType\_sexp>}

Opens a system dialog for opening a document and returns the selected path by <documentPath\_svar>. If <documentPath\_svar> returns an empty string (""), the operation was not successful. If an Array\$[] is used multiselection is possible. If the selection in this case is failed, the first array item returns an empty string ("").

An existing start path will be defined by <startPath\_sexp>. An empty string let the document browser start at the default data path. If this attribute is not used the start begins at the last visited location.

To shrink the number of select-able documents use <mimeType\_sexp> to specify the wanted types.

Unlike Adoc.open, **only** document providers specified for **reading** can be selected.

The possibility of choices is usually larger.

The returned URI will only be safe for read access.

See also Adoc.open

**Adoc.name** <documentName\_svar>, <documentPath\_sexp>

Returns the human-friendly name of the document by the given document path. If this is not provided then the name should default to the last segment of the documents's URI. If an error occurs an empty string "" is returned.

**Adoc.size** <documentSize\_nvar>, <documentPath\_sexp>

Returns the number of bytes in the document identified by the openable document path. Gives back -1 if unknown.

### **Adoc.mimetype <success\_nvar>, <mimeType\_svar>, <documentPath\_sexp>**

Returns 1 if the MIME type of the document specified by the document path was returned successfully. Otherwise 0 is returned. The value given by <documentMimeType\_nvar> returns the MIME type of this document.

### **Adoc.lastmodified <success\_nvar>, <documentLastModified\_nvar>, <documentPath\_sexp>**

Returns 1 if the last modification of the document specified by the document path was returned successfully. Otherwise 0 is returned. The value given by <documentLastModified\_nvar> returns the time when this document was last modified, measured in milliseconds since January 1st, 1970, midnight GMT.

Example:

```
FILE.ROOT pp$ , "_SourceSamples"
ADOC.PATH dp$,"file://" + pp$ + "/" + "f01_commands.bas"
ADOC.LASTMODIFIED success, lm, dp$
? USING("", "%TF %tT" ,lm, lm) % Returns "2019-06-30 21:01:01"
```

### **Adoc.delete <success\_nvar>, <documentPath\_sexp>**

Returns 1 if the document specified by the document path was deleted successfully. Otherwise 0 is returned.

Min. KitKat 4.4 (API 19) is needed.

### **Adoc.rename <success\_nvar>, <documentPath\_sexp>, <newName\_sexp>**

Returns 1 if the document specified by the document path was renamed specified by <newName\_sexp> successfully. Otherwise 0 is returned.

Min. Lollipop 5.0 (API 21) is needed.

Is the API 19 or 20 copy the document and delete the first document by Adoc.delete.

### **Adoc.write <success\_nvar>, <documentPath\_sexp>, <newContent\_sexp>{, <charSet\_sexp>}, <grant\_perm\_nexp>}**

Returns 1 if the document specified by the document path was completely filled with a String content specified by <newContent\_sexp> successfully. Otherwise 0 is returned.

Note: The specified document has to be created before!

For binary data specify the <charSet\_sexp> with the "\_ISO-8859-1" character set instead of the default "\_UTF-8".

A required permission is granted until it is revoked or the app deinstalled.

You can choose between following character sets: "\_US-ASCII", "\_UTF-8", "\_UTF-16", "\_UTF-16BE", "\_UTF-16LE" and "\_ISO-8859-1".

Min. KitKat 4.4 (API 19) is needed.

To take control over the grant of a permission <grant\_perm\_nexp> is used. If a Document Provider does not ask for permissions use 0 for false. Default is 1 for true.

### **Adoc.write.file <success\_nvar>, <documentPath\_sexp>, <filePath\_sexp>{, <grant\_perm\_nexp>}**

Returns 1 if the document specified by the document path was completely filled with a file content specified by <filePath\_sexp> successfully. Otherwise 0 is returned.



Note: The specified document has to be created before!

To take control over the grant of a permission <grant\_perm\_nexp> is used. If a Document Provider does not ask for permissions use 0 for false. Default is 1 for true.

**Adoc.read <success\_nvar>, <result\_sexp>, <documentPath\_sexp>{, <charSet\_sexp>, <grant\_perm\_nexp>}**

Returns 1 if the document specified by the document path gives the full String content specified by <result\_sexp> successfully back. Otherwise 0 is returned.

For binary data specify the <charSet\_sexp> with the "\_ISO-8859-1" character set instead of the default "\_UTF-8".

A required permission is granted until it is revoked or the app deinstalled.

You can choose between following character sets: "\_US-ASCII", "\_UTF-8", "\_UTF-16", "\_UTF-16BE", "\_UTF-16LE" and "\_ISO-8859-1".

Min. KitKat 4.4 (API 19) is needed.

To take control over the grant of a permission <grant\_perm\_nexp> is used. If a Document Provider does not ask for permissions use 0 for false. Default is 1 for true.

**Adoc.read.file <success\_nvar>, <filePath\_sexp>, <documentPath\_sexp>{, <grant\_perm\_nexp>}**

Returns 1 if the document specified by the document path gives the full file content specified by <filePath\_sexp> successfully back. Otherwise 0 is returned.

To take control over the grant of a permission <grant\_perm\_nexp> is used. If a Document Provider does not ask for permissions use 0 for false. Default is 1 for true.

**Adoc.grab <result\_sexp>, <documentPath\_sexp>{, <charSet\_sexp>}**

Puts the full String content specified by the <documentPath\_sexp> into <result\_sexp>.

For binary data specify the <charSet\_sexp> with the "\_ISO-8859-1" character set instead of the default "\_UTF-8".

A required permission is granted until it is revoked or the app deinstalled.

You can choose between following character sets: "\_US-ASCII", "\_UTF-8", "\_UTF-16", "\_UTF-16BE", "\_UTF-16LE" and "\_ISO-8859-1".

Normally you will use this instead of Adoc.read, because you can read also only for reading specified content providers.

Keep in mind, that in this case a broken network connection is not checked.

**Adoc.exists <lvar>, <documentPath\_sexp>{, <grant\_perm\_nexp>}**

Reports if the <documentPath\_sexp> directory or document exists. If the directory or document does not exist, the <lvar> will contain zero. If the file or directory does exist, the <lvar> will be returned as non-zero.

A required permission is granted until it is revoked or the app deinstalled.

Min. KitKat 4.4 (API 19) is needed.

To take control over the grant of a permission <grant\_perm\_nexp> is used. If a Document Provider does not ask for permissions use 0 for false. Default is 1 for true.

**Adoc.revoke <success\_nvar>, <documentPath\_sexp>**

Returns 1 if the document's access permission specified by the document path was deleted successfully. Otherwise 0 is returned.

Min. KitKat 4.4 (API 19) is needed.





## Provider

Android supports **document** and file providers.

A Provider is in context of Android a tool, that can be used independent of a running app.

These can be used to share files over the borders of Android's Scoped Storage.

Manages documents and exposes them to the Android system for sharing.

Providers are running in an independent instance if it will be called by an internal or external request.

That means, this independent instance will be executed if your app is running or not.

Also if you stop the running at the System App Info Preferences.

And that is also the reason that you can control the providers only indirectly over Shared Preferences from your app.

Unfortunately, this provides a large back door for unwanted data and hidden manipulation, as this instance also has access to all public static methods of the internal Java code.

Particularly dangerous if "true" is entered under AndroidManifest.xml <Provider android:exported = "true"

But that is the case if you follow Google's advice and you provide documents under Scoped Storage by a Documents (Adoc) Provider or any other Provider for public use.

To prevent this issue the command PROVIDER offers appropriate switches / settings.

As already mentioned and according to Google's advice, you should use providers starting with scoped storage at the latest.

Are there alternatives?

Get Rid of Providers:

???? vv

~~- But you have also the possibility to use the Download directory with free access from all apps for data transfer. History repeats itself with a different name. Nowadays the Download directory is the WORK directory of the 80's. It is not secure so you have to encrypt the sensitive data, but in this case you know you are in a tank of sharks! Delete also encrypted sensitive data in the Download directory as soon as possible. Note that when you uninstall your app you will not have write access to your old files in the future, starting with Scoped Storage (Android 11 at the latest).~~

???? ^^

- If you have data with roughly less one Mb in size you can use the data transfer over intents.

Maybe little zipping will help.

- A simple nice internal file picker is more secure than a provider for internal use.

- A simple ftp(s) server could do the task also if you are in a network or your device creates a WLAN hotspot.

In case of using a Provider, a Adoc Provider is the best choice of security.

But not a solution if you want to provide a file maybe for a PDF viewer or mail client. In this case the File Provider is the choice.

## Provider <bundle\_pointer\_nexp>

Sets permissions and values. These settings are stored as default values and used at start as an independent provider called by a the same or different app.

If you provide a variable that is not a valid Bundle pointer, the command creates a new Bundle and returns the Bundle pointer in your variable. Otherwise it writes into the Bundle your variable or expression points to.

The bundle keys and possible values are in the table below:

Key	Type	Value
<b>_FileP_Read</b>	numeric	Sets the read permission if > 0 within the File Provider. Default is 0.
<b>_FileP_Write</b>	numeric	Sets the write permission if > 0 within the File Provider. Default is 0. If other apps should be give write permissions these apps should use a <b>full</b> exception handling.
<b>_FileP_Dir</b>	numeric	Sets the get-directory permission if > 0 within the File Provider. Default is 0. FileProviderDir.txt returns the updated directory content. FileProviderDateDir.txt returns the updated directory content with a timestamp at the begin of each record. If it is 0 both files return an empty string. Note, that <b>_FileP_Read</b> have to be > 0 also.
<b>_FileP_DirOut</b>	numeric	If > 0, directories will be returned also. Default = 0
<b>_ADocP_Read</b>	numeric	Sets the read permission if > 0 within the ADoc Provider. Default is 0.
<b>_ADocP_Write</b>	numeric	Needs additionally also permissions from the Google Play Store.
<b>_ADocP_Dir</b>	numeric	Sets the get-directory permission if > 0 within the ADoc Provider. Default is 0. ADocProviderDir.txt returns the updated directory content. ADocProviderDateDir.txt returns the updated directory content with a timestamp at the begin of each record. If it is 0 both files return an empty string. Note, that <b>_ADocP_Read</b> have to be > 0 also.
<b>_ADocP_DirOut</b>	numeric	If > 0, directories will be returned also. Default = 0
<b>_ADoc_Summary</b>	String	Summary text under app name and icon. Default is "".

File Providers support only read and write files. OliBasic supports also the recursive directory content by reading out FileProviderDir.txt and FileProviderDateDir.txt.

The file path of the File Provider is:

```
Program.info blnf
```

```
Bundle.get blnf, "_PackageName", pn$
```

```
"content://" + pn$ + "/" + yourFilePath$
```

Note: yourFilePath\$ is the path within the File Provider directory.

ADoc Providers support the bunch of the ADoc commands. OliBasic supports also the recursive directory content by reading out ADoc ProviderDir.txt and ADoc ProviderDateDir.txt.

The root directory of the ADoc Provider is:

```
Program.info blnf
```

```
Bundle.get blnf, "_PackageName", pn$
```

```
"content://" + pn$ + ".documents/document/" + "root" + "%3A" + yourDocumentPath$
```

Note: yourDocumentPath\$ is the path within the ADoc Provider directory.

To get the content of a File provided by a File Provider use

```
Adoc.read success, result$, "content://com.my.app" + "/" + "test.txt", , 0
```

In case of OliBasic itself:

```
File.root iPath$, "_FileProvider"
```

```
Byte.open w, ftp, iPath$ + "/" + "testText.txt"
```

```
Byte.write.buffer ftp, "Basic4ever\nFile Provider"
```

```
Byte.close ftp
```

```
Adoc.read success, result$, "content://com.rfo.basicOli" + "/" + "testText.txt", , 0
```

```
PRINT success, result$
```

In case of your compiled app replace the package name from OliBasic to yours like "content://com.rfo.basicOli" into "content://com.my.newapp".

## Bluetooth

BASIC! implements Bluetooth in a manner which allows the transfer of data bytes between an Android device and some other device (which may or may not be another Android device).

Before attempting to execute any BASIC! Bluetooth commands, you should use the Android "Settings" Application to enable Bluetooth and pair with any device(s) with which you plan to communicate.

When Bluetooth is opened using the **Bt.open** command, the device goes into the Listen Mode. While in this mode it waits for a device to attempt to connect.

For an active attempt to make a Bluetooth connection, you can use the Connect Mode by successfully executing the **Bt.connect** command. Upon executing the **Bt.connect** command the person running the program is given a list of paired Bluetooth devices and asked. When the user selects a device, BASIC! attempts to connect to it.

You should monitor the state of the Bluetooth using the **Bt.status** command. This command will report states of Listening, Connecting and Connected. Once you receive a "Connected" report, you can proceed to read bytes and write bytes to the connected device.

You can write bytes to a connected device using the **Bt.write** command.

Data is read from the connected device using the **Bt.read.bytes** command; however, before executing **Bt.read.bytes**, you need to find out if there is data to be read. You do this using the **Bt.read.ready** command.

Once connected, you should continue to monitor the status (using **Bt.status**) to ensure that the connected device remains connected.

When you are done with a particular connection or with Bluetooth in general, execute **Bt.close**.

The sample program, f35\_bluetooth, is a working example of Bluetooth using two Android devices in a "chat" type application.

### **Bt.open** {{{0|1}, <delimiter\_nexp>}, <del\_nexp>}

Opens Bluetooth in Listen Mode. If you do not have Bluetooth enabled (using the Android Settings Application) then the person running the program will be asked whether Bluetooth should be enabled. After **Bt.open** is successfully executed, the code will listen for a device that wants to connect.

The optional parameter determines if BT will listen for a secure or insecure connection. If no parameter is given or if the parameter is 1, then a secure connection request will be listened for. Otherwise, an insecure connection will be listened for. It is not possible to listen for either a secure or insecure connection with one **Bt.open** command because the Android API requires declaring a specific secure/insecure open.

If **Bt.open** is used in graphics mode (after **Gr.open**), you will need to insert a **Pause 500** statement after the **Bt.open** statement.

In most cases, the counterpart of a Bluetooth connection uses a line feed at the end of a message.

The bytes of a message will be stored in memory until the character code of <delimiter\_nexp> (greater than -1) is reached. Default is 10 that is equal to a line feed / newline.

After receiving the delimiter the content of the memory will be returned by **Bt.read.bytes** or **Bt.utf\_8.read.bytes**. Should the delimiter also be deleted <del\_nexp> has to be greater than 0. Default is 1.

If you try to deal with long messages (> 128 bytes) the sender should send two line feeds at the end off a message if you deal with microcontrollers. Because the Serial Port Profile

(SPP) maximum payload capacity is device-dependent from 128 to 1024 bytes. Note that UTF-8 characters can have one or two bytes. It seems that connections between Android devices split the messages larger than about 400 bytes as needed.

The maximum bit rate of a serial service for a Bluetooth connection should be 115200 bits/s. See also <https://electronics.stackexchange.com/questions/203763/bluetooth-module-throughput-uart-baud-rate-how-fast-is-it>.

To get the old command characteristic set <delimiter\_nexp> to -1.

## Bt.close

Closes any previously opened Bluetooth connection. Bluetooth will automatically be closed when the program execution ends.

## Bt.connect {0|1}

Commands BASIC! to connect to a particular device. Executing this command will cause a list of paired devices to be displayed. When one of these devices is selected the **Bt.status** will become "Connecting" until the device has connected.

The optional parameter determines if BT will seek a secure or insecure connection. If no parameter is given or if the parameter is 1, then a secure connection will be requested. Otherwise, an insecure connection will be requested.

## Bt.disconnect

Disconnects from the connected Bluetooth device and goes into the Listen status. This avoids having to use **Bt.close** + **Bt.open** to disconnect and wait for a new connection.

## Bt.reconnect

This command will attempt to reconnect to a device that was previously connected (during this Run) with **Bt.connect** or a prior **Bt.reconnect**. The command cannot be used to reconnect to a device that was connected following a **Bt.open** or **Bt.disconnect** command (i.e. from the **Listening** status).

You should monitor the Bluetooth status for **Connected** (3) after executing **Bt.reconnect**.

## Bt.status {{<connect\_var>},{, <name\_svar>},{, <address\_svar>}}

Gets the current Bluetooth status and places the information in the return variables. The available data are the current connection status (in <connect\_var>), and the friendly name and MAC address of your Bluetooth hardware (in <name\_svar> and <address\_svar>). All parameters are optional; use commas to indicate omitted parameters (see Optional Parameters).

If the connection status variable <connect\_var> is present, it may be either a numeric variable or a string variable. The table shows the possible return values of each type:

Numeric Value	String Value	Meaning
-1	Not enabled	Bluetooth not enabled
0	Idle	Nothing going on
1	Listening	Listening for connection
2	Connecting	Connecting to another device
3	Connected	Connected to another device
4	Lost	Connection lost and try again

5	Failed	Getting a connection failed and try again
---	--------	---

If the device name string variable <name\_svar> is present, it is set to the friendly device name. If your device has no Bluetooth radio, the string will be empty.

If the address string variable <address\_svar> is present, it is set to the MAC address of your Bluetooth hardware, represented as a string of six hex numbers separated by colons: "00:11:22:AA:BB:CC".

### OnBtStatus:

Interrupt label that traps if the status of the bluetooth connection has changed. BASIC! executes the statements following the **OnBtStatus:** label until it reaches a **Bt.onStatus.resume**.

### Bt.onStatus.resume

Resumes execution at the point in the BASIC! program where the **OnBtStatus:** interrupt occurred.

### Bt.write {<exp> {,|;}} ...

### Bt.utf\_8.write {<exp> {,|;}} ...

Writes data to the Bluetooth connection.

If the comma (,) separator is used then a comma will be printed between the values of the expressions.

If the semicolon (;) separator is used then nothing will separate the values of the expressions.

If the semicolon is at the end of the line, the output will be transmitted immediately, with no newline character(s) added.

The parameters are the same as the **Print** parameters. This command is essentially a **Print** to the Bluetooth connection, with two differences:

- Only one byte is transmitted for each character; the upper byte is discarded. Binary data and ASCII text are sent correctly, but Unicode characters may not be.  
If you need the full Unicode character set use **Bt.utf\_8.write**.
- A line that ends with a semicolon is sent immediately, with no newline character(s) added.

This command with no parameters sends a newline character to the Bluetooth connection.

### Bt.read.ready <nvar>

Reports in the numeric variable the number of messages ready to be read. If the value is greater than zero then the messages should be read until the queue is empty.

### OnBtReadReady:

Interrupt label that traps the arrival of a message received on the Bluetooth channel (see "Interrupt Labels"). If a Bluetooth message is ready (**Bt.read.ready** would return a non-zero value) BASIC! executes the statements after the **OnBtReady:** label, where you can

read and handle the message. When done, execute the **Bt.onReadReady.Resume** command to resume the interrupted program.

### **Bt.onReadReady.resume**

Resumes execution at the point in the program where it was interrupted by the Bluetooth Read Ready event.

### **Bt.read.bytes <svar>**

### **Bt.utf\_8.read.bytes <svar>**

The next available message is placed into the specified string variable. If there is no message then the string variable will be returned with an empty string (""). Each message byte is placed in one character of the string; the upper byte of each character is 0. This is similar to **Byte.read.buffer**, which reads binary data from a file into a buffer string.

If you need the full Unicode character set use **Bt.utf\_8.read.bytes**.

### **Bt.device.name <svar>**

Returns the name of the connected device in the string variable. A run-time error will be generated if no device (Status <> 3) is connected.

### **Bt.set.UUID <sexp>**

A Universally Unique Identifier (UUID) is a standardized 128-bit format for a string ID used to uniquely identify information. The point of a UUID is that it's big enough that you can select any random 128-bit number and it won't clash with any other number selected similarly. In this case, it's used to uniquely identify your application's Bluetooth service. To get a UUID to use with your application, you can use one of the many random UUID generators on the web.

Many devices have common UUIDs for their particular application. The default BASIC! UUID is the standard Serial Port Profile (SPP) UUID: "00001101-0000-1000-8000-00805F9B34FB".

You can change the default UUID using this command.

Some information about 16-bit and 128-bit UUIDs can be found at:

<http://farwestab.wordpress.com/2011/02/05/some-tips-on-android-and-bluetooth/>



## BLE

### **Ble.open**

Opens the BLE system.

### **Ble.close**

Closes the BLE system.

### **Ble.scan <flag\_nexp>**

Enables(1) / disables(0) BLE scanning.

### **Ble.devices Array\$[]**

Returns found nearby devices in form of addresses like XX:XX:XX:XX:XX:XX for found devices.

### **Ble.close**

### **Ble.status <device\_sexp>, <name\_svar>**

Returns the device name for a given device address.

### **Ble.rssi <device\_sexp>, <rssi\_nvar>**

Returns the RSSI for a given device address.

### **Ble.connect <device\_sexp>**

Connects to a found device.

### **Ble.disconnect**

Disconnects from the device.

### **Ble.status <status\_svar>**

Gets the connection status ("Disconnected", "Scanning", "Connecting", "Discovering", "Connected").

### **Ble.services Array\$[]**

Returns services on a connected device.

### **Ble.characteristics <service\_sexp>, Array\$[]**

Returns characteristics on a connected device for a given service.

### **Ble.notify <char\_sexp>, <flag\_nexp>**

Enables(1) / disables(0) notifications for a given characteristic.

### **Ble.write <char\_sexp>, <data\_sexp>**

Writes data to a device characteristic.

### **Ble.read.request <char\_sexp>**

Request a read for given characteristic.

### **Ble.read <data\_svar>**

Read characteristic (empty if not yet received).

## USB

The Universal Serial Bus enables data transfer via serial communication.

Supported drivers:

- FTDI FT232 (I am not going to brick your device, trust me 😊, Felipe Herranz)
- Silicon Labs CP210x
- Prolific PL2303HX (at least HX version)
- CH340/CH341
- Generic CDC driver

The serial communication has several protocol variants.

Known issue: If you want to finish the USB service or after a connection lost, you have to save your variables and execute the RUN() function.

If you run into trouble, you can try a micro controller and compiling this code.

Arduino IDE example:

```
void setup() {
  Serial.begin(9600);
  pinMode(LED_BUILTIN, OUTPUT);
}
void loop() {
  delay(1000);
  Serial.println("OUT");
  while(Serial.available() > 0) {
    digitalWrite(LED_BUILTIN, HIGH);
    delay(100);
    digitalWrite(LED_BUILTIN, LOW);
    Serial.print("I received: ");
    Serial.println(Serial.read(), DEC);
  }
}
```

### Usb.devices deviceId\_Array[], deviceDescrip\_Array\$[]

Returns arrays with select-able USB devices. The first numeric array contains the device IDs the second String array the device descriptions. Note that also mass storage will be returned.

### Usb.open <success\_svar>{, <bundle\_nexp>}

Opens an USB channel specified by the Bundle <bundle\_nexp> and returns success or an error.

If more than one channel can be used for serial communication, the user can optionally be asked for the right one.

The bundle keys and possible values are in the table below:

Key	Type	Value
<b>_BaudRate</b>	numeric	Sets the baud rate. Common are 30, 50, 75, 110, 130, 300, 600, 1200, 2400, 4800, <b>9600</b> , 14400, 19200, <b>38400</b> , 57600, <b>115200</b> , 128000, 230400, 250000, 256000, 500000, 512000, 921600, 1000000, 2000000 etc. in example. Keep in mind, that this BASIC interpreter is limited in speed. Default is <b>115200</b> bits per second.
<b>_DataBits</b>	numeric	Sets the count of data bits. Available are 5, 6, 7 or 8. Default is <b>8</b> .
<b>_StopBits</b>	numeric	Sets the count of stop bits. Available are 1, 1.5 or 2. Default is <b>1</b>
<b>_Parity</b>	String	Sets the parity selectable are <b>_None</b> , . Default is <b>_None</b> .
<b>_FlowControl</b>	String	Sets flow control selectable are <b>_Off</b> , <b>_Xon_Xoff</b> (Maybe, currently not tested.), <b>_Dsr_Dtr</b> or <b>_Rts_Cts</b> . The last two only for CP2102 and FT232 usable. Default is <b>_Off</b> .
<b>_Pause</b>	numeric	Some Arduinos would need some sleep because firmware wait some time to know whether a new sketch is going to be uploaded or not. In this case are 2000 milliseconds a candidate for a first try. Default is <b>0</b> .
<b>_Delimiter</b>	String	Different Programs, Micro controller, Terminals etc. are able to send also different codes for a new line. Mostly a LF terminates a line or message. Some terminals use CR LF like CHR\$(13) + CHR\$(10). For control codes like CR, LF or TAB(9) you have to use the CHR\$() function. Default is CHR\$(10) ( <b>LF</b> ).
<b>_CharSet</b>	String	If you need to transfer characters from 0 to 255 maybe for binary data, you can choose the " <b>_ISO-8859-1</b> " character set instead of the default " <b>_UTF-8</b> ". This bundle key can be changed during data transfer operations also.
<b>_Base64</b>	numeric	The data transfer will be encoded by the Bas64 encoding algorithm if the numeric argument is greater than 0. Both character sets handle Bas64 correctly. This bundle key can be changed during data transfer operations also. Default is <b>0</b> .

### OnUsbReadReady:

Interrupt label that traps if data via USB arrived. BASIC! executes the statements following the **OnUsbReadReady:** label until it reaches a **Html.onHtmlReturn.resume**.

### Usb.onReadReady.resume

Resumes execution at the point in the BASIC! program where the **OnUsbReadReady:** interrupt occurred.

### Usb.read.bytes <sval>

Reads a String of bytes via USB. Depended on the buffer size and other circumstances the number of bytes is limited. A delimiter like the default Linefeed character CHR\$(10) ("
") initiates a "Have data to read!" message. This message triggers an interrupt detectable by onUsbReadReady:. Note, If the delimiter is "", each packet can be read as it is.

Example:

```
USB.devices d[], i$[]
Array.length al, d[]
IF al > 1 & d[1] > -1 % If more than one device can be selected
  sel = 0 : Dialog.single retBut, sel, i$[], "Serial Devices", "OK", "CHANCEL"
ENDIF
Bundle.Put MB, "_BaudRate", 9600
IF sel > 0 & retBut = 1 THEN Bundle.Put MB, "_DeviceID", d[sel] : ?d[sel]
USB.open mB

onUsbReadReady:
  USB.read.bytes b$
  b$ = REPLACE$(b$, CHR$(13), "") // Removes each Carriage Return if any.
  PRINT B$
  USB.write "Test"
  USB.onreadready.resume
```

### Usb.write <sexp>

Writes an UTF-8 encoded String via USB by default. The data transfer can also be Base64 encoded or by the ISO-8859-1 character set.

### OnUsbStatus:

Interrupt label that traps if status message via USB arrived. BASIC! executes the statements following the **OnUsbStatus:** label until it reaches a **Usb.onReadReady.resume**.

### Usb.onStatus.resume

Resumes execution at the point in the BASIC! program where the **OnUsbReadReady:** interrupt occurred.

### Usb.Status <svar>

Returns the current status message.

Possible are:

- An USB device is attached
- USB service started
- USB Ready
- USB Permission granted
- CTS change
- DSR change
- USB Permission not granted
- USB not supported
- No USB device connected

- USB device not working
- CDC driver not working
- USB device not supported
- Current USB device detached. To reconnect restart by RUN()
- An other USB device is detached

### Run {{<filename\_sexp> }, <data\_sexp>}

This command will terminate the running of the current program and then load and run the BASIC! program named in the filename string expression. The filename is relative to BASIC's "source/" directory. If the filename is "program.bas" and your <pref base drive> is "/sdcard" (the default), then the file "/sdcard/rfo-basic/source/program.bas" will be executed.

The optional data string expression provides for the passing of data to the next program. The passed data can be accessed in the next program by referencing the special variable, **##\$**.

**Run** programs can be chained. A program loaded and run by means of the **Run** command can also run another program file. This chain can be as long as needed.

When the last program in a **Run** chain ends, tapping the BACK key will display the original program in the BASIC! Editor.

When a program ends with an error, the Editor tries to highlight the line where the error occurred. If the program with the error was started by a **Run** command, the Editor does not have that program loaded. Any highlighting that may be displayed is meaningless.

If the first parameter = "" or not given, a program with the current file path will load and run (if not compiled maybe with different code). In APK mode the App will be relaunched.

Note: "file://" + <full\_path\_svar> can now be used for absolute filepaths.

IF <filename\_sexp> = "" or is not defined, the current program itself is restarting again.

Be carefully for not to be caught in an endless loop.

Example:

```
! Run "Files.bas"
! Run "Files.bas", ""
! Run %Endless loop?
! Run "" %Endless loop?
! Run " " %File Not found
PROGRAM.INFO bdi
BUNDLE.GET bdi, "BasName", BasName$
PRINT BasName$
FILE.ROOT aPath$, "_Source"
IF ##$ <> "1"
! Run , "1"

! RUN "file://" + aPath$ + "/" + BasName$, "1"
ENDIF
! Run "../source/Files.bas", ""
FILE.ROOT dp$, "_Source"
? dp$

RUN "file://" + dp$ + "/" + "Files.bas"
```

GitHub#215

## BigDecimal

In BASIC! numbers are normally stored as values from type Double. For precise calculation the BigDecimal engine is our choice. The best way to store these values lossless in BASIC! is using Strings.

You do not need always BigDecimals instead of Double. For addition and subtraction you can use also the exponential shifting hack in the range of the bounds of Double numbers.

$k = 10^{21}$

$c = (a*k + b*k) / k$  or  $c = (a*k - b*k) / k$

$10^{21}$  is the useful maximum. If you want three correct digits behind the decimal point use  $10^5$ . **But with Double you get maximal only 15 correct digits.**

Source for PI: <http://www.pibel.de>

Source for e: <http://www.gutenberg.org/files/127/127.txt>

### BigD.add <result\_svar>, <first\_sexp>, <second\_sexp>

Returns a new BigDecimal as a String whose value is <first\_sexp> + <second\_sexp>. The scale of the result is the maximum of the scales of the two arguments.

### BigD.sum <result\_svar>, Array\$[]

Returns a new BigDecimal as a String whose value is the sum of all array String items. The scale of the result is the maximum of the scales of all arguments.

### BigD.subtract <result\_svar>, <first\_sexp>, <second\_sexp>

Returns a new BigDecimal as a String whose value is <first\_sexp> - <second\_sexp>. The scale of the result is the maximum of the scales of the two arguments.

### BigD.multiply <result\_svar>, <first\_sexp>, <second\_sexp>

Returns a new BigDecimal as a String whose value is <first\_sexp> \* <second\_sexp>. The scale of the result is the sum of the scales of the two arguments.

### BigD.divide <result\_svar>, <first\_sexp>, <second\_sexp>, <scale\_nexp>, <roundingMode\_sexp>

Returns a new BigDecimal as a String whose value is <first\_sexp> / divisor <second\_sexp>. As scale of the result the parameter scale is used. If rounding is required to meet the specified scale, then the specified rounding mode <roundingMode\_sexp> is applied.

See also BigD.round for rounding details.

### BigD.remainDividing <integral\_svar>, <remainder\_svar>, <first\_sexp>, <second\_sexp>

Returns a BigDecimal Strings which contain the integral part of <first\_sexp> / divisor <second\_sexp> as <integral\_svar> and the remainder <first\_sexp> - <first\_sexp>/int(divisor) \* divisor as <remainder\_svar>.

See also MOD()

## MOD(<nexp1>, <nexp2>)

Returns the remainder of <nexp1> divided by <nexp2>. If <nexp2> is 0, the function generates a runtime error **returns NaN**.

## BigD.abs <result\_svar>, <first\_sexp>

Returns a BigDecimal as a String whose value is the absolute value of <first\_sexp>. The scale of the result is the same as the scale of <first\_sexp>.

See also ABS()

## BigD.frac <result\_svar>, <first\_sexp>

Returns a new BigDecimal as a String whose value is the fractional part of <first\_sexp>. The scale of the result is the same as the scale of <first\_sexp>.

## FRAC(<nexp>)

Returns the fractional part of <nexp>. 3.4 becomes 0.4 and -3.4 becomes -0.4.

**FRAC(n)** is equivalent to "n – INT(n)".

**FRAC(n)** is four times slower than the equivalent, but so exact as possible.

## BigD.int <result\_svar>, <first\_sexp>

Returns a new BigDecimal as a String whose value is the integral part of <first\_sexp>.

See also INT(), ROUND()

## INT(<nexp>)

Returns the integer part of <nexp>. 3.X becomes 3 and -3.X becomes -3. This operation may also be called truncation, rounding down, or rounding toward zero.

So the decimal point and the digits behind him will be deleted.

The int() function in most other BASIC dialects works different. See ROUND() for more details.

See also INT(), FRAC(), ROUND()

## BigD.compare <result\_nvar>, <first\_sexp>, <second\_sexp>

Compares BigDecimal <first\_sexp> with <second\_sexp>. Returns one of the three values 1, 0, or -1. The method behaves as if <first\_sexp> - <second\_sexp> is computed. If this difference is > 0 then 1 is returned, if the difference is < 0 then -1 is returned, and if the difference is 0 then 0 is returned. This means, that if two decimal instances are compared which are equal in value but differ in scale, then these two instances are considered as equal.

## BigD.equals <result\_nvar>, <first\_sexp>, <second\_sexp>

Returns 1.0 if <first\_sexp> and <second\_sexp> are BigDecimal instances and equal otherwise <result\_svar> returns 0.0. Two big decimals are equal if their unscaled value **and** their scale is equal. For example, 1.0 (10\*10<sup>-1</sup>) is not equal to 1.00 (100\*10<sup>-2</sup>).

Similarly, zero instances are not equal if their scale differs.



### **BigD.toDouble <result\_nvar>, <first\_sexp>**

Returns <first\_sexp> BigDecimal as a double value. NaN (Not a Number), Double.POSITIVE\_INFINITY or Double.NEGATIVE\_INFINITY are not supported and thrown a runtime error.

Note, that if the unscaled value has more than 53 significant digits, then this decimal cannot be represented exactly in a double variable. In this case the result is rounded.

### **BigD.FromDouble <result\_svar>, <number\_nexp>**

Returns a new BigDecimal as a String from the Double <number\_nexp>.

NaN (Not a Number), Double.POSITIVE\_INFINITY or Double.NEGATIVE\_INFINITY are not supported and thrown a runtime error.

### **BigD.toBase <result\_svar>, <string\_sexp>, <base\_sexp>**

Returns a new BigDecimal as a String whose value is the result of the String <string\_sexp> encoded by the base "\_Bin", "\_Oct" or "\_Hex"(default, if wrong also).

### **BigD.FromBase <result\_svar>, <string\_sexp>, <base\_sexp>**

Returns a new BigDecimal as a String whose value is the result of the encoded String <string\_sexp> decoded by the base "\_Bin", "\_Oct" or "\_Hex"(default, if wrong also).

### **BigD.hashCode <result\_svar>, <first\_sexp>**

Returns a hash code as a String for <first\_sexp> as a BigDecimal.

The hash code is computed as

$s[0]*31^{(n-1)} + s[1]*31^{(n-2)} + \dots + s[n-1]$

using int arithmetic, where  $s[i]$  is the  $i$ th character of the string,  $n$  is the length of the string, and  $^$  indicates exponentiation. The hash value of an empty string is zero.

### **BigD.max <result\_svar>, <first\_sexp>, <second\_sexp>**

Returns a new BigDecimal as a String whose value is the maximum of <first\_sexp> and <second\_sexp> as BigDecimal.

See also MAX()

### **BigD.min <result\_svar>, <first\_sexp>, <second\_sexp>**

Returns a new BigDecimal as a String whose value is the minimum of <first\_sexp> and <second\_sexp> as BigDecimal.

See also MIN()

### **BigD.movePointLeft <result\_svar>, <first\_sexp>, <n\_nexp>**

Returns a new BigDecimal instance as a String where the decimal point has been moved  $n$  places to the left. If  $n < 0$  then the decimal point is moved  $-n$  places to the right.

The result is obtained by changing its scale. If the scale of the result becomes negative, then its precision is increased such that the scale is zero.

Note, that movePointLeft with  $n=0$  returns a result which is mathematically equivalent, but which has scale  $\geq 0$ .

### BigD.movePointRight <result\_svar>, <first\_sexp>, <n\_nexp>

Returns a new BigDecimal instance as a String where the decimal point has been moved  $n$  places to the right. If  $n < 0$  then the decimal point is moved  $-n$  places to the left.

The result is obtained by changing its scale. If the scale of the result becomes negative, then its precision is increased such that the scale is zero.

Note, that movePointRight with  $n=0$  returns a result which is mathematically equivalent, but which has  $\text{scale} \geq 0$ .

### BigD.pow <result\_svar>, <first\_sexp>, <n\_nexp>

Returns a new BigDecimal as a String whose value is <first\_sexp> raised to the <n\_nexp> power  $\rightarrow \text{<first_sexp>}^n$ . The scale of the result is  $n * \text{the scale from <first_sexp>}$ .

BigD.pow  $x$  \$, 0.0,  $r$  \$ returns "1", even if  $x$  \$ = "0".

Implementation Note: The implementation is based on the ANSI standard X3.274-1996 algorithm.

Operation Note:  $n$  have to be in the bounds of 0 to 999999999.

See also POW()

### BigD.sqr <result\_svar>, <first\_sexp>, <scale\_nexp>

Returns a new BigDecimal as a String whose value is the closest approximation of the positive square root of <first\_sexp>. If the value of <first\_sexp> is negative, the function generates a runtime error. The maximum scale is given by <scale\_nexp>.

See also: BigD.scale, SQR()

### BigD.precision <result\_nvar>, <first\_sexp>

Returns the precision as a Double of <first\_sexp> as BigDecimal. The precision is the number of decimal digits used to represent this decimal. It is equivalent to the number of digits of the unscaled value. The precision of 0 is 1 (independent of the scale).

### BigD.round <result\_svar>, <first\_sexp>, <scale\_nexp>, <roundingMode\_sexp>

Returns a new BigDecimal as a String whose value is <first\_sexp>, rounded according to scale and rounding mode. As scale of the result the parameter scale is used. If rounding is required to meet the specified scale, then the specified rounding mode <roundingMode\_nexp> is applied.

There are **eight** rounding modes:

Mode:	Meaning:	-3.8	-3.5	-3.1	-3.0	3.0	3.1	3.5	3.8
"HD"	Half-down	-4.0	-3.0	-3.0	-3.0	3.0	3.0	3.0	4.0
"HE"	Half-even	-4.0	-4.0	-3.0	-3.0	3.0	3.0	4.0	4.0
"HU"	Half-up	-4.0	-4.0	-3.0	-3.0	3.0	3.0	4.0	4.0
"D"	Down	-3.0	-3.0	-3.0	-3.0	3.0	3.0	3.0	3.0
"U"	Up	-4.0	-4.0	-4.0	-3.0	3.0	4.0	4.0	4.0
"F"	Floor	-4.0	-4.0	-4.0	-3.0	3.0	3.0	3.0	3.0
"C"	Ceiling	-3.0	-3.0	-3.0	-3.0	3.0	4.0	4.0	4.0
"LI"	Legacy int()	-4.0	-4.0	-4.0	-3.0	3.0	3.0	3.0	3.0

In this table, "down" means "toward zero" and "up" means "away from zero" (toward  $\pm\infty$ )

In most cases is "Half-Up"  $\rightarrow$  "HU" the best choice.

See also ROUND()

## ROUND(<value\_nexp>{, <scale\_nexp>{, <roundingMode\_sexp>}})

In its simplest form, **ROUND(<value\_nexp>)**, this function returns the closest whole number to <nexp>. You can use the optional parameters to specify more complex operations.

The <scale\_nexp> is an optional decimal place count. It sets the number of places to the right of the decimal point. The last digit is rounded. ~~The decimal place count must be  $\geq 0$ .~~ If <scale\_nexp> is < 0 only a faster Half-up rounding mode is used. Omitting the parameter is the same as setting it to zero.

The <roundingMode\_sexp> is an optional rounding mode. It is a one- or two-character mnemonic code that tells **ROUND()** what kind of rounding to do. It is not case-sensitive. There are eight rounding modes:

Mode:	Meaning:	-3.8	-3.5	-3.1	-3.0	3.0	3.1	3.5	3.8
"HD"	Half-down	-4.0	-3.0	-3.0	-3.0	3.0	3.0	3.0	4.0
"HE"	Half-even	-4.0	-4.0	-3.0	-3.0	3.0	3.0	4.0	4.0
"HU"	Half-up	-4.0	-4.0	-3.0	-3.0	3.0	3.0	4.0	4.0
"D"	Down	-3.0	-3.0	-3.0	-3.0	3.0	3.0	3.0	3.0
"U"	Up	-4.0	-4.0	-4.0	-3.0	3.0	4.0	4.0	4.0
"F"	Floor	-4.0	-4.0	-4.0	-3.0	3.0	3.0	3.0	3.0
"C"	Ceiling	-3.0	-3.0	-3.0	-3.0	3.0	4.0	4.0	4.0
"LI"	Legacy int()*	-4.0	-4.0	-4.0	-3.0	3.0	3.0	3.0	3.0

In this table, "down" means "toward zero" and "up" means "away from zero" (toward  $\pm\infty$ )

"Half" refers to behavior when a value is half-way between rounding up and rounding down (x.5 or -x.5). "Half-down" rounds x.5 towards zero and "half-up" rounds x.5 away from zero.

"Half-even" is either "half-down" or "half-up", whichever would make the result **even**. 4.5 and 3.5 both round to 4.0. "Half-even" is also called "banker's rounding", because it tends to average out rounding errors.

In most cases is **"Half-Up" → "HU"** the best choice.

**\*Legacy int() → "LI"** is compatible to the most other BASIC dialects

Examples of Most Other BASIC Dialects	Examples of BASIC!
myNumber = Int(99.8) ' Returns 99.	myNumber = round(99.8, 0, "LI") % Returns 99.
myNumber = Fix(99.8) ' Returns 99.	myNumber = INT(99.8) % Returns 99.
myNumber = Int(-99.8) ' Returns -100.	myNumber = round(-99.8, 0, "LI") % Returns -100.
myNumber = Fix(-99.8) ' Returns -99.	myNumber = INT(-99.8) % Returns -99.
myNumber = Int(-99.2) ' Returns -100.	myNumber = round(-99.2, 0, "LI") % Returns -100.
myNumber = Fix(-99.2) ' Returns -99.	myNumber = INT(-99.2) % Returns -99.

So BASIC's **INT(<nexp>)** works like the **Fix()** command in the table above.

If you do not provide a `<roundingMode_sexp>`, **ROUND()** adds +0.5 and rounds down (toward zero). This is a [bequest behavior, copied](#) from earlier versions of BASIC!. **ROUND(n)** is NOT the same as **ROUND(n, 0)**.

**ROUND()** generates a runtime error if `<count_nexp> < 0` or `<mode_sexp>` is not valid.

Examples:

```
pi = ROUND(3.14159)           % pi is 3.0
pi = ROUND(3.14159, 2)       % pi is 3.14
pi = ROUND(3.14159, , "U")   % pi is 4.0
pi = ROUND(3.14159, 4, "F")  % pi is 3.1415
negpi = ROUND(-3.14159, 4, "D") % negpi is -3.1416
```

Note that **FLOOR(n)** is exactly the same as **ROUND(n, 0, "F")**, but **FLOOR(n)** is a little faster. In the same way, **CEIL(n)** is the same as **ROUND(n, 0, "C")**, and **INT(n)** is the same as **ROUND(n, 0, "D")**.

## FORMAT\$(`<pattern_sexp>`, `<nexp>/<sexp>`)

Returns a string with `<nexp>` or `<sexp>` formatted by the pattern `<pattern_sexp>`.

Keep in mind, that this function **does not round** the given number. [BigDecimal numbers](#) are stored as a String and can used directly.

<b>Leading Sign</b>	A negative (-) character for numbers < 0 or a space for numbers >= 0. The Sign and the Floating Character together form the <b>Floating Field</b> .
<b>Floating Character</b>	If the first character of the pattern is not "#" or "." or "-" then that character becomes a "floating" character. This pattern character is typically a "\$". If no floating character is provided then a space character is used. See also <b>Overflow</b> , below.
<b>Decimal Point</b>	The pattern may have one optional decimal point character ("."). If the pattern has no decimal point, then only the whole number is output. Any digits that would otherwise appear after the decimal point are not output.
<b># Character</b> (before decimal, or no decimal)	Each "#" is replaced by a digit from the number. If there are more "#" characters than digits, then the leading "#" character(s) are replaced by <b>space(s)</b> .
<b># Character</b> (after decimal point)	Each "#" is replaced by a digit from the number. If there are more "#" characters than significant digits, then the trailing "#" character(s) are replaced by <b>zero(s)</b> . The number of "#" characters after the pattern decimal point specifies the number of decimal digits that will be output.
<b>% Character</b> (before decimal, or no decimal)	Each "%" is replaced by a digit from the number. If there are more "%" characters than digits, then the leading "%" character(s) are replaced by <b>zero(s)</b> .
<b>% Character</b> (after decimal)	The "%" character is not allowed after the decimal point. This is a syntax error.
<b>Non-pattern Characters</b>	If any pattern character (other than the first) is not # or %, then that character is copied directly into the output. If the character would appear before the first digit of the number, it is replaced by a space.

	This feature is usually used for commas.
<b>Overflow</b>	If the number of digits exceeds the number of # and % characters, then the output has the ** characters inserted in place of the Floating Field.
<b>Output Size</b>	The number of characters output is always the number of characters in the pattern plus one for the sign plus one more for the space if the pattern has no Floating Character.

## Notes

The sign and the floating character together form a **Floating Field** two characters wide that always appears just before the first digit of the formatted output. If there are any leading spaces in the formatted output, they are placed before the floating field.

The "#" character generates leading spaces, not leading zeros. "##.###" formats 0.123 as ".123". If you want a leading zero, use a "%". For example "%.###", "%#.###", or "##%" all assure a leading zero.

Be careful mixing # and % characters. Doing so except as just shown can produce unexpected results.

The number of characters output is always the number of characters in the pattern plus the two floating characters.

Examples:

Function Call	Output	Width
Format\$( "##.###,###", 1234567)	<b>1,234,567</b>	12 characters
Format\$( "%%,%%%,%%%.#", 1234567.89)	<b>01,234,567.8</b>	14 characters
Format\$( "\$###.###", 123456)	<b>\$123,456</b>	9 characters
Format\$( "\$###.###", -1234)	<b>-\$1,234</b>	9 characters
Format\$( "\$###.###", 12)	<b>\$12</b>	9 characters
Format\$( "\$%%,%%%,%%%.%", -12)	<b>-\$000,012</b>	9 characters
Format\$( "##.#", 0)	<b>.0</b>	6 characters
Format\$( "%#.#", 0)	<b>0.0</b>	6 characters
Format\$( "\$###.###", -1234.5)	<b>**234.50</b>	8 characters

```
t = -15.97
mFormatString$ = "%#.#"
where = Is_In(".", mFormatString$)
mFrac$ = MID$(mFormatString$, where + 1)
result$ = FORMAT$( mFormatString$, ROUND(t, LEN(mFrac$)))
PRINT result$
```

## BigD.scale <result\_nvar>, <first\_sexp>

Returns the scale of <first\_sexp> BigDecimal as a Double. The scale is the number of digits behind the decimal point. The value of <first\_sexp> BigDecimal is the unsignedValue \* 10<sup>-scale</sup>. If the scale is negative, then <first\_sexp> BigDecimal represents a big integer.

## BigD.sign <result\_nvar>, <first\_sexp>

Returns the signum function of the BigDecimal value of <first\_sexp> as a Double, representing its sign.

Returns

-1 if BigDecimal of <first\_sexp> < 0,

0      if BigDecimal of <first\_sexp> = 0,  
1      if BigDecimal of <first\_sexp> > 0.  
See also SIGN()

### BigD.nanoTime <result\_svar>

Returns the current timestamp of the most precise timer available on the local system, in nanoseconds. Equivalent to Linux's CLOCK\_MONOTONIC.

This timestamp should only be used to measure a duration by comparing it against another timestamp on the same device. Values returned by this method do not have a defined correspondence to wall clock times; the zero value is typically whenever the device last booted. Use BigD.time if you want to know what time it is.

To compare two nano Time values

```
BigD.nanoTime tic1$
```

```
...
```

```
BigD.nanoTime tic2$
```

One should use BigD.compare result, tic2\$, tic1\$, because of the possibility of numerical overflow.

### BigD.time <result\_svar>

Returns the current time in milliseconds since January 1, 1970 00:00:00.0 UTC.

This method always returns UTC times, regardless of the system's time zone. This is often called "Unix time" or "epoch time". ~~Use a java.text.DateFormat instance to format this time for display to a human.~~

This method shouldn't be used for measuring timeouts or other elapsed time measurements, as changing the system time can affect the results. Use BigD.nanoTime for that.

See also TIME()

### BigD.date <result\_svar>, <first\_sexp>

Returns the time in milliseconds since January 1, 1970 00:00:00.0 UTC.

Note, only for Android

Example:

```
mDate$ = "2020-10-15T09:27:37Z+0100"
```

```
BigD.date result$, mDate$
```

### BigD.toEngineering <result\_svar>, <first\_sexp>

Returns a string representation of <first\_sexp> BigDecimal. This representation always prints all significant digits of this value.

If the scale is negative or if  $\text{scale} - \text{precision} \geq 6$  then engineering notation is used.

Engineering notation is similar to the scientific notation except that the exponent is made to be a multiple of 3 such that the integer part is  $\geq 1$  and  $< 1000$ .

### BigD.toScientific <result\_svar>, <first\_sexp>

Returns a canonical string representation of <first\_sexp> BigDecimal. If necessary, scientific notation is used. This representation always prints all significant digits of this value.

If the scale is negative or if  $\text{scale} - \text{precision} \geq 6$  then scientific notation is used.

### BigD.ulp <result\_svar>, <first\_sexp>

Returns the unit in the last place (ULP) of <first\_sexp> BigDecimal instance. An ULP is the distance to the nearest big decimal with the same precision.

The amount of a rounding error in the evaluation of a floating-point operation is often expressed in ULPs. An error of 1 ULP is often seen as a tolerable error.

For class BigDecimal, the ULP of a number is simply 10-scale. For example,

BigD.ulp "123", r\$ returns "1"

BigD.ulp "1.23", r\$ returns "0.01"

## SHELL Command

### Shell <result\_svar>, <command\_sexp>

Opens a **SHELL** to execute system commands <command\_sexp>. The working directory is in **opposite** to **System.open** set to "**root**". The command is waiting for a result <result\_svar>.

Example:

```
FILE.ROOT path$  
d$ = "cat " + path$ + "/" + "htmldemo1.html"  
SHELL r$, d$  
PRINT r$
```



## TCP/IP Sockets

TCP/IP Sockets provide for the transfer of information from one point on the Internet to another. There are two genders of TCP/IP Sockets: Servers and Clients. Clients must talk to Servers. Servers must talk to Clients. Clients cannot talk to Clients. Servers cannot talk to Servers.

Every Client and Server pair have an agreed-upon protocol. This protocol determines who speaks first and the meaning and sequence of the messages that flow between them.

Most people who use a TCP/IP Socket will use a Client Socket to exchange messages with an existing Server with a predefined protocol. One simple example of this is the Sample Program file, **f31\_socket\_time.bas**. This program uses a TCP/IP client socket to get the current time from one of the many time servers in the USA.

A TCP/IP Server can be set up in BASIC!; however, there are difficulties. The capabilities of individual wireless networks vary. Some wireless networks allow servers. Most do not. Servers can usually be run on WiFi or Ethernet Local Area Networks (LAN).

If you want to set up a Server, the way most likely to work is to establish the Server inside a LAN. You will need to provide Port tunneling (forwarding) from the LAN's external Internal IP to the device's LAN IP. You must be able to program (setup) the LAN router in order to do this.

Clients, whether running inside the Server's LAN or from the Internet, should connect to the LAN's external IP address using the pre-established, tunneled Port. This external or WAN IP can be found using:

Graburl ip\$, "<http://icanhazip.com>"

This is not the same IP that would be obtained by executing **Socket.myIP** on the server device.

Note: The specified IPs do not have to be in the numeric form. They can be in the name form.

The Sample Program, **f32\_tcp\_ip\_sockets.bas**, demonstrates the socket commands for a Server working in conjunction with a Client. You will need two Android devices to run this program.

On Android devices the default transfer character set is the character set of the filesystem here UTF-8.

If you need to transfer characters from 0 to 255 maybe for binary data, you can choose the "\_ISO-8859-1" character set instead of the default "\_UTF-8".

But the `Socket.Server.write.file`, `Socket.Server.read.file`, `Socket.Client.write.file` or `Socket.Client.read.file` commands in conjunction with "\_ISO-8859-1" do not work as expected, because sending and receiving of `Chr$(65535)` is not possible.

An ephemeral port is a communications endpoint (port) of a transport layer protocol of the Internet protocol suite that is used for only a short period of time for the duration of a communication session. Such short-lived ports are allocated automatically within a predefined range of port numbers by the IP stack software of a computer operating system.

How to choose one?

The RFC 6056 says that the range for ephemeral ports should be 1024–65535.

IANA and RFC 6335 suggests the range 49152–65535 for dynamic or private ports.

Older Windows versions and BSD use ports 1025–5000 as ephemeral ports.



Many Linux kernels use the port range 32768–60999.  
Android use a Linux kernel, thus this port range is recommended.

Four-digit ports such as 2021 with the ending 21 (ftp) or 8080 with the ending 80 (http) are also common for permanent connections.

## TCP/IP Client Socket Commands

**Socket.client.connect** <server\_sexp>, <port\_nexp> {{ , <wait\_lexp> }, <char\_set\_sexp> }

Create a Client TCP/IP socket and attempt to connect to the Server whose Host Name or IP Address is specified by the Server string expression using the Port specified by Port numeric expression.

The optional "wait" parameter determines if this command waits until a connection is made with the Server. If the parameter is absent or true (non-zero), the command will not return until the connection has been made or an error is detected. If the Server does not respond, the command should time out after a couple of minutes, but this is not certain.

If the parameter is false (zero), the command completes immediately. Use

**Socket.client.status** to determine when the connection is made. If you monitor the socket status, you can set your own time-out policy. You must use the **Socket.client.close** command to stop a connection attempt that has not completed.

Using the optional <charset\_sexp> you can choose between following character sets: "\_ISO-8859-1" and "\_UTF-8"(default).

**Socket.client.read.byte** <svar>

Read a byte from the previously-connected Server and place the byte into the string variable. To avoid an infinite delay waiting for the Server to send a line, the

**Socket.client.read.ready** command can be repeatedly executed with timeouts.

### What About Reading Unknown Number of Bytes?

The best answer is that your application either needs to know beforehand how many bytes to expect, or the "application protocol" needs to somehow tell it how many bytes to expect ... or when all bytes have been sent. Possible approaches are:

- The application protocol uses fixed message sizes
- The application protocol message sizes are specified in message headers
- The application protocol uses end-of-message markers
- The application protocol is not message based, and the other end closes the connection to say "that's the end".

```
line$ = ""  
DO  
  SOCKET.CLIENT.READ.BYTE mByte$  
  line$ = line$ + mByte$  
UNTIL mByte$ = CHR$(10) | mByte$ = CHR$(13)  
line$ = REPLACE$(line$, CHR$(10), "")  
line$ = REPLACE$(line$, CHR$(13), "")
```

Is the same as

```
SOCKET.CLIENT.READ.LINE line$
```

```
byteAsNumber = 0 ... 65535  
out$ = CHR$(byteAsNumber )  
SOCKET.SERVER.WRITE.BYTES out$
```

```
SOCKET.CLIENT.READ.BYTE in$  
byteAsNumber = UCODE(in$)
```

### **Socket.client.read.line** <line\_svar>

Read a line from the previously-connected Server and place the line into the line string variable. The command does not return until the Server sends a line. To avoid an infinite delay waiting for the Server to send a line, the **Socket.client.read.ready** command can be repeatedly executed with timeouts.

Note, the end of the line is detected if a CR or LF is receiving.

## **TCP/IP Server Socket Commands**

### **Socket.server.create** <port\_nexp>{, <char\_set\_sexp>}

Establish a Server that will listen to the Port specified by the numeric expression, <port\_nexp>.

Using the optional <charset\_sexp> you can choose between following character sets: "\_ISO-8859-1" and "\_UTF-8"(default).

### **Socket.server.read.byte** <svar>

Read a byte sent from the previously-connected Client and place the byte into the string variable <svar>. To avoid an infinite delay waiting for the Client to send a line, the **Socket.server.read.ready** command can be repeatedly executed with timeouts. See also Socket.client.read.byte.

### **Socket.server.read.line** <svar>

Read a line sent from the previously-connected Client and place the line into the string variable <svar>. The command does not return until the Client sends a line. To avoid an infinite delay waiting for the Client to send a line, the **Socket.server.read.ready** command can be repeatedly executed with timeouts.

Note, the end of the line is detected if a CR or LF is receiving.

## UDP Socket Commands

User Datagram Protocol is a simpler message-based connectionless protocol. Connectionless protocols do not set up a dedicated end-to-end connection. Communication is achieved by transmitting information in one direction from source to destination without verifying the readiness or state of the receiver.

**Unreliable** – When an UDP message is sent, it cannot be known if it will reach its destination; it could get lost along the way. There is no concept of acknowledgment, retransmission, or timeout.

**Not ordered** – If two messages are sent to the same recipient, the order in which they arrive cannot be predicted.

**Lightweight** – There is no ordering of messages, no tracking connections, etc. It is a small transport layer designed on top of IP.

**Datagrams** – Packets are sent individually and are checked for integrity only if they arrive. Packets have definite boundaries which are honored upon receipt, meaning a read operation at the receiver socket will yield an entire message as it was originally sent.

**No congestion control** – UDP itself does not avoid congestion. Congestion control measures must be implemented at the application level.

**Broadcasts** – being connectionless, UDP can broadcast - sent packets can be addressed to be receivable by all devices on the subnet.

Source: Wikipedia

**Port range** – from 32768 to 60999 is recommended.

A transfer request is always initiated targeting port 69, but the data transfer ports are chosen independently by the sender and receiver during the transfer initialization.

**If you want a read and write communication (IoT), make sure that there is a pause of maybe 500ms for interaction.**

### UDP.read <result\_svar>, <port\_nexp>, <wait\_nexp> {, <char\_set\_sexp>}

Listens for an UDP message on port <port\_nexp> and read a string if any datagram message arrived. The command waits <wait\_nexp> milliseconds. If <wait\_nexp> is 0 it waits infinite. Using this command in a loop with 1200 ms for the first try is recommended. The character set can be specified by <char\_set\_sexp>. Use "\_UTF-8" (default) for text and "\_ISO-8859-1" for binary data. An input until 65527 bytes is accepted.

65,535 bytes (8-byte header + 65,527 bytes of data) ????

Example:

```
DO
DO
    UDP.READ r$, 54321, 1200, "_ISO-8859-1"
UNTIL r$ <> ""
PRINT r$
UNTIL 0
```

**UDP.write <message\_svar>, <ip\_adress\_sexp>, <port\_nexp> {, <char\_set\_sexp>}**

Writes a string <message\_svar> as an UDP datagram message to a local ip adress <ip\_adress\_sexp> on port <port\_nexp>. The character set can be specified by <char\_set\_sexp>. Use "\_UTF-8" (default) for text and "\_ISO-8859-1" for binary data. An output until 65507 bytes is accepted.

Example:

```
m$= "My message! "  
FOR i = 1 TO 100  
  mm$ = m$ + STR$(i)  
  UDP.WRITE mm$, "192.168.1.12", 54321, "_ISO-8859-1"  
  PAUSE 50  
NEXT
```

## TFTP Socket Client

TFTP is a simple protocol for transferring files, implemented on top of the UDP/IP protocols using well-known port number 69. TFTP was designed to be small and easy to implement, and therefore it lacks most of the advanced features offered by more robust file transfer protocols. TFTP only reads and writes files from or to a remote server. It cannot list, delete, or rename files or directories and it has no provisions for user authentication. Today TFTP is generally only used on local area networks (LAN).

A transfer request is always initiated targeting port 69, but the data transfer ports are chosen independently by the sender and receiver during the transfer initialization.

Source: Wikipedia

**Tftp.get <remote\_host\_sexp>, <remote\_file\_sexp>, <local\_file\_sexp>{{{, <mode\_nexp>}, <timeout\_nexp>}, <ok\_svar>}**

Gets the remote file <remote\_file\_sexp> from the remote host <remote\_host> and save it into a local file <local\_file\_sexp>. Note that an existing file will be overwritten.

The transfer mode is specified by the optional <mode\_nvar>. The binary mode is chosen if <mode\_nvar> is > 0. If it is 0 the Netascii, an 8-bit extension of the 7-bit ASCII character space, is used. Default is 1.

The optional <timeout\_nvar> in milliseconds stops the command if the data transfer failed without an error. Default is 60000.

If an error occurs <ok\_svar> returns an exception message. Is the message empty the data transfer was successful.

**Tftp.put <remote\_host\_sexp>, <remote\_file\_sexp>, <local\_file\_sexp>{{{, <mode\_nexp>}, <timeout\_nexp>}, <ok\_svar>}**

Uploads the specified source file <local\_file\_sexp> to the specified destination file <remote\_file\_sexp> on the connected Tftp server <remote\_host\_sexp>.

Note in most cases an existing file will be overwritten on the Tftp server.

The transfer mode is specified by the optional <mode\_nvar>. The binary mode is chosen if <mode\_nvar> is > 0. If it is 0 the Netascii, an 8-bit extension of the 7-bit ASCII character space, is used. Default is 1.

The optional <timeout\_nvar> in milliseconds stops the command if the data transfer failed without an error message. Default is 60000.

If an error occurs <ok\_svar> returns an exception message. Is the message empty the data transfer was successful.

## NFC Commands

NFC (Near Field Communication) refers to the possibility of contactless transmission of data between two devices or a so-called tag. With NFC, data can be transferred between an active device (e.g. Android smartphone or tablet with NFC) and a passive NFC tag, small amounts of data (e.g. a few 100 bytes of payload) over distances of a few centimeters. The NFC tag is powered by the active NFC device via electromagnetic induction during transmission. Radio waves at 13.56 MHz are used for data transmission.

Source: [https://www.droidwiki.org/wiki/Near\\_Field\\_Communication](https://www.droidwiki.org/wiki/Near_Field_Communication)

Please check if your device is able to read and write NFC-Tags. Take care that a spare battery support NFC. The manufacturer's original is the best choice.

After tests:

Three tested SAMSUNG devices support NFC fine. A Nokia 5.1 with Android (One) 10 does not work although specified.

If you do not have Bluetooth enabled (using the Android Settings Application) then the person running the program will be asked whether NFC should be enabled.

**The following commands support only NFC cards and tags.**

### NFC.read <bundle\_nexp>

The bundle pointer <bundle\_nexp> specifies a timer and returns the results of a NFC card or NFC tag.

The bundle keys and possible values are in the table below:

Key	Type	Value
<b>_AskForNfc</b>	numeric	Time in milliseconds to switch on if asked for NFC because it is not enabled. If 0, no question is asked. Default is 10000.
<b>_Timer</b>	numeric	Specifies a timer, which returns after given milliseconds to the main program. Default is 0. In this case it will be returned only after reading a card or tag. Using a timer is strongly recommended.
<b>_Error</b>	String	Returns an error in the event of an error, otherwise an empty string.
<b>_Id</b>	String	Returns the ID of the card or tag. Note, this ID is not in all cases unique, because you can buy cards and tags with specified ID series.
<b>_NfcData</b>	String	Returns all NDEF text data records as a CSV string delimited by a LF (CHR\$(10) or \n). If the string is UTF-8 encoded, it can be decoded by DECODE\$("ASCII", ...\$)
<b>_Records</b>	Array of String	Returns an Array of NDEF text data records. If the records are UTF-8 encoded, it can be decoded by DECODE\$("ASCII", ...\$)
<b>_Tag</b>	String	Returns the possible NFC types supported by the card or tag delimited by ", " (comma and space).

Note that URIs, AppIDs and MIME records cannot be read because Android catches them first. For this issue the app NFC Tools (app id = com.wakdev.wdnfc) by wakdev.com is recommended.

Example:

```
BUNDLE.PUT nfcBun, "_Timer", 5000
NFC.READ nfcBun
DEBUG.ON
DEBUG.DUMP.BUNDLE nfcBun
BUNDLE.GET nfcBun, "_Records", records$[]
DEBUG.DUMP.ARRAY records$[]
```

### NFC.write <bundle\_nexp>

The bundle pointer <bundle\_nexp> returns the results of an error and sets write settings of the NFC card or NFC tag.

The bundle keys and possible values are in the table below:

Key	Type	Value
<b>_AskForNfc</b>	numeric	Time in milliseconds to switch on if asked for NFC because it is not enabled. If 0, no question is asked. Default is 10000.
<b>_Timer</b>	numeric	Specifies a timer, which returns after given milliseconds to the main program. Default is 0. In this case it will be returned only after writing a card or tag. Using a timer is strongly recommended. In case of writing it should be equal or greater than 8000 milliseconds.
<b>_Error</b>	String	Returns an error in the event of an error, otherwise an empty string. If the writing fails no error is thrown. Please read the tag again to see that the writing is ok.
<b>_NfcData</b>	String	Submits one NDEF text data record as a string, if _Records is not specified. The string is UTF-8 encoded.
<b>_Records</b>	Array of String	Submits an Array of NDEF text data records to write. The records are UTF-8 encoded.
<b>_Types</b>	Array of String	Submits an Array of NDEF types. Default is "" (" _Text"). Other types are " _App" and " _Uri". " _App" expects an AppId like "com.rfo.basicOli" " _Uri" expects an uri beginning with http(s)://..., file///..., content://... If the array is shorter than the array in _Records the unspecified are from type "" (" _Text").
<b>_MessageLength</b>	numeric	Returns the message length in Bytes.
<b>_TagSpace</b>	numeric	Returns the available NDEF memory space in Bytes.

**Writing with Android devices, mostly smartphones, is not an easy task, because reading is fast, but when you write you need to be patient. Please put your card or tag on a desk and move your device down slowly.**

If the writing fails and reading returns not the wished result you have the opportunity to use the app NFC Tools (app id = com.wakdev.wdnfc) by wakdev.com to fix it by writing a simple text record (after a formatting if needed) or copying an other tag.

NFC.write is not able to overwrite \_App or \_Uri records. In this case, NFC Tools is also your helper when overwriting by a simple sentence.

**To say it again handle carefully your device at card or tag writing. Not in a hurry. Success needs slow motion. Otherwise your tag can be destroyed forever.**  
Cards and tags with the NTAG 213 or NTAG 216 are recommended.

Example:

```
ARRAY.LOAD records$, "First data record", "Second data record"  
BUNDLE.PUT nfcBun, "_Timer", 8000  
NFC.WRITE nfcBun  
BUNDLE.GET nfcBun, "_Error", error$  
IF error$ THEN PRINT error$
```



**Input {<prompt\_sexp>}, <result\_var>{, {<default\_exp>}{, <canceled\_nvar>}}**

Generates a dialog box with an input area and an **OK** button. When the user taps the button, the value in the input area is written to the variable <result\_var>.

The <prompt\_sexp> will become the dialog box title. If the prompt expression is empty ("") or omitted, the dialog box will be drawn without a title area.

If the return variable <result\_var> is numeric, the input must be numeric, so the only key taps that will be accepted are 0-9, "+", "-", and ".". If <result\_var> is a string variable, the input may be any string.

If a <default\_exp> is given then its value will be placed into the input area of the dialog box. The default expression type must match the <result\_var> type.

The variable <canceled\_nvar> controls what happens if the user cancels the dialog, either by tapping the BACK key or by touching anywhere outside of the dialog box.

If you provide a <canceled\_nvar>, its value is set to **false** (0) if the user taps the **OK** button, and **true** (1) if the users cancels the dialog.

If you do not provide a <canceled\_nvar>, a canceled dialog is reported as an error. Unless there is an "OnError:" the user will see the messages:

Input dialog cancelled  
Execution halted

If there is an "OnError:" label, execution will resume at the statement following the label.

The <result\_var> parameter is required. All others are optional. These are all valid:

INPUT "prompt", result\$, "default", isCanceled  
INPUT , result\$, "default"  
INPUT "prompt", result\$, , isCanceled  
INPUT "prompt", result\$  
INPUT , result\$

Note the use of commas as parameter placeholders (see Optional Parameters).

Note also, that in some Android versions in Graphic Mode the width of the box is shrinking according to the text size. Some spaces at the end of the prompt will help if needed.

**Dialog.Cust.Open <layout\_sexp>{{{, <bgColor\_sexp>}, <bgImage\_sexp>}, <scroll\_nexp>}**

Opens the dialog description process. The layout is specified by <layout\_sexp>.

If it is not a standard layout like a picker each layout has up to 100 static IDs, which can be switched to be shown dynamically.

The optional background color is set by <bgColor\_sexp>.

{Alpha,}Red,Green,Blue

(comma delimited string)

or

\_{Alpha,}ColorName

({comma delim.} string)

or

#{hn}hn nhn

(hex. string)

An optional background image file can be set by <bgImage\_sexp>.

A scroll bar can be enabled by <scroll\_nexp> if it is > 0. Default is 1.

Table of layouts	
Layout	Description
<b>_TimePicker</b>	Shows a system time picker. Returns a bundle named _TimePicker. This bundle returns the keys _Hour and _Minute.
DIALOG.CUST.OPEN "_TimePicker" DIALOG.CUST.CALL retBut, retBundle, "", "Time Picker", "OK", "Cancel" BUNDLE.GB retBundle, "_TimePicker", tpBundle BUNDLE.GET tpBundle, "_Hour", hour BUNDLE.GET tpBundle, "_Minute", minute PRINT RIGHT\$("0"+INT\$(hour), 2); ":"; RIGHT\$("0"+INT\$(minute), 2)	
<b>_DatePicker</b>	Shows a system date picker. Returns a bundle named _DatePicker. This bundle returns the keys _Year, _Month and _DayOfMonth.
DIALOG.CUST.OPEN "_DatePicker" DIALOG.CUST.CALL retBut, retBundle, "", "Date Picker", "OK", "Skip" BUNDLE.GB retBundle, "_DatePicker", dpBundle BUNDLE.GET dpBundle, "_Year", year % year\$ is also possible BUNDLE.GET dpBundle, "_Month", month % month\$ is also possible BUNDLE.GET dpBundle, "_DayOfMonth", dayOfMonth PRINT INT\$(year); "-"; RIGHT\$("0"+INT\$(month), 2); "-"; RIGHT\$("0"+INT\$(dayOfMonth), 2)	
<b>_Calendar</b>	Shows a system calendar. Returns a bundle named _Calendar. This bundle returns the key _Date. The date is the time since 1970-01-01 00:00 in milliseconds. The date can be predefined using the key _SetDate in the layout bundle in the following DIALOG.CUST.CALL command.

Table of layouts	
Layout	Description
	BUNDLE.PUT bndPtr, "_SetDate", INT\$(TIME(2022, 12, 25, 12, 0, 0)) DIALOG.CUST.OPEN "_Calendar" DIALOG.CUST.CALL retBut, retBundle, "", "Calendar", "OK", "Skip", "", bndPtr BUNDLE.GB retBundle, "_Calendar", cBundle BUNDLE.GET cBundle, "_Year", year % year\$ is also possible BUNDLE.GET cBundle, "_Month", month % month\$ is also possible BUNDLE.GET cBundle, "_DayOfMonth", dayOfMonth PRINT INT\$(year); "-"; RIGHT\$("0"+INT\$(month), 2); "-"; RIGHT\$("0"+INT\$(dayOfMonth), 2)
<b>_AnalogClock</b>	Shows a system analog clock.
	DIALOG.CUST.OPEN "_AnalogClock" retBut = -4000 % Turns off the clock in 4 seconds DIALOG.CUST.CALL retBut, retBundle, "", "Analog Clock", "OK"
<b>_Login</b>	Shows a login form. 1 ImageView, 2 EditText [User] 3 ImageView, 4 EditText [Password]
	DIALOG.CUST.OPEN "_Login" DIALOG.CUST.IMAGE 1, "cartman.png" DIALOG.CUST.EDIT 2, "", "User", "_Text", 1, 30, "", "_Blue" DIALOG.CUST.IMAGE 3, "galaxy.png" DIALOG.CUST.EDIT 4, "", "Password", "_TextPassword", 1, 30, "", "_Red" DIALOG.CUST.CALL retBut, retBundle, "", "Please login!", "Ok", "Cancel" ! If the returned result of an EditText is a number, a number variable can also be used. BUNDLE.GET retBundle, "_Edit 2", user\$ BUNDLE.GET retBundle, "_Edit 4", password\$ PRINT user\$, password\$
<b>_EditRecord</b>	Shows an edit form. 1 TextView (max. 50) 2 EditText (max. 50) 3 T...
<b>_ImageEditRecord</b>	Shows an edit form. 1 TextView (max. 33) 2 ImageView (max. 33) 2 EditText (max. 33) 4 T...
<b>_Dialog0</b>	Dummy layout
<b>_Dialog1</b>	Customizable layout
<b>_Dialog2</b>	Customizable layout
<b>_Dialog3</b>	Customizable layout
<b>_Dialog4</b>	Customizable layout
<b>_Dialog5</b>	Customizable layout
<b>_Dialog6</b>	Customizable layout
<b>_Dialog7</b>	Customizable layout
<b>_Dialog8</b>	Customizable layout
<b>_Dialog9</b>	Customizable layout
<b>_Dialog10</b>	Customizable layout
<b>_Dialog11</b>	Customizable layout

**Dialog.cust.text <ID\_nval>, <text\_sexp>{{{, <textSize\_nexp>}, <textStyle\_sexp>}, <textColor\_sexp>}**

Switches a given TextView to show at position <ID\_nval> in the specified layout.

The optional <textSize\_nexp> sets the text size. Default are 12 dpi of the Android standard resolution of 160 dpi, which are converted into the current screen resolution automatically.

The optional <textStyle\_nexp> sets the text stile. Available are "\_Normal", "\_Bold", "\_Italic" and "\_Bold\_Italic".

The optional <textColor\_nexp> sets the text color. Default is the standard color of the chosen style in the Dialog.cust.call layout bundle.

Example:

```
DIALOG.CUST.TEXT 1, "Name", 16, "_Bold", "_Red"
```

**Dialog.cust.image <ID\_nval>, <image\_sexp>{{{, <imageHeight\_nexp>}, <newSvgColor\_sexp>}, <oldSvgColor\_nexp>}**

Switches a given ImageView to show at position <ID\_nval> in the specified layout.

The file path of the image has to be set by <image\_sexp>.

The optional <imageHeight\_nexp> sets the image height. Default are 48 dpi of the Android standard resolution of 160 dpi, which are converted into the current screen resolution automatically. The size ratio of the image is retained. To get the default height of 48 dpi use 0. The original size can be choosen if <imageHeight\_nexp> is -1.

If the image is stored as a vector SVG file, you can change its color by <newSvgColor\_sexp>. The color to replace is specified by <oldSvgColor\_nexp>. Default is "#333333".

Use the OliBasic color notations like:

{Alpha,}Red,Green,Blue  
(comma delimited string)

or

\_{Alpha,}ColorName  
({comma delim.} string)

or

{#hn}hnhnhn  
(hex. string)

Example:

```
DIALOG.CUST.IMAGE 2, "cartman.png"
```

**Dialog.cust.edit <ID\_nval>, <text\_sexp>, <hint\_sexp>, <inputType\_sexp>{{{, <enabled\_nvar>}, <textSize\_nexp>}, <textStyle\_sexp>}, <textColor\_sexp>}**

Switches a given EditText to show at position <ID\_nval> in the specified layout.

A given text have to be set by <text\_sexp>.

A hint in the background of the edit file is to specify by <hint\_sexp>. It is strongly recommended in case of using within the landscape orientation.

Following input type options are available:

"\_None",  
"\_Text",  
"\_TextCapCharacters", "\_TextCapWords", "\_TextCapSentences", "\_TextAutoCorrect",  
"\_TextAutoComplete", "\_TextMultiLine", "\_TextImeMultiLine", "\_TextNoSuggestions",  
"\_TextUri", "\_TextEmailAddress", "\_TextEmailSubject", "\_TextShortMessage",  
"\_TextLongMessage", "\_TextPersonName", "\_TextPostalAddress", "\_TextPassword",  
"\_TextVisiblePassword", "\_TextWebEditText", "\_TextFilter", "\_TextPhonetic",  
"\_TextWebEmailAddress", "\_TextWebPassword",  
"\_Number",  
"\_NumberSignedDecimal", "\_NumberSigned", "\_NumberDecimal", "\_NumberPassword"  
"\_Date", "\_Datetime", "\_Time"

If <enabled\_nvar> is greater than 0 the input field can be edited. Default is 1.

The optional <textSize\_nexp> sets the text size. Default are 12 dpi of the Android standard resolution of 160 dpi, which are converted into the current screen resolution automatically.

The optional <textStyle\_nexp> sets the text stile. Available are "\_Normal", "\_Bold", "\_Italic" and "\_Bold\_Italic".

The optional <textColor\_nexp> sets the text color. Default is the standard color of the chosen style in the Dialog.cust.call layout bundle.

The results are returned by <retBnd\_nval> of the Dialog.cust.call command.

Example:

DIALOG.CUST.EDIT 3, "Joel Smith", "Name", "\_Text", 1, 30, "\_Normal", "\_Red"

**Dialog.cust.call <retBut\_nval>, <retBnd\_nval>, <message\_sexp>, <title\_sexp>{{{, <button1\_sexp>}, <button2\_sexp>}, <button3\_sexp>}, <bndPtr\_nexp>}**

Finishes the dialog description process and starts the dialog. The keystrokes are returned by <retBut\_nval>. If <retBut\_nval> is assigned a negative number before the command is called, a timer is set and this counts down in milliseconds before it ends the dialog.

The results will be returned by the bundle <retBnd\_nval>. See Dialog.cust.open for details. A message can be set using <message\_sexp>. Be careful when using it. Test your dialog in landscape format as well.

The title can be set by <title\_sexp>.

The buttons will be optionally set by <button1\_sexp>, <button2\_sexp> and <button3\_sexp>.

More layout specifications and predefined values can be set by the bundle <bndPtr\_nexp>.

Table of options		
Key	Value	Description
<b>_Style</b>	<b>_Default</b>	Default theme
	<b>_Dark</b>	Dark theme and Autosize mode
	<b>_Bright</b>	Bright theme and Autosize mode
<b>_Icon</b>	String	Filename of a header icon
<b>_PositionH</b>	<b>_Left</b>	
	<b>_Center</b>	Default
	<b>_Right</b>	
<b>_PositionV</b>	<b>_Top</b>	
	<b>_Center</b>	Default
	<b>_Bottom</b>	
<b>_TitleSize</b>	numeric	
<b>_TitleColor</b>	{Alpha,}Red,Green,Blue (comma delimited string) or _{Alpha,}ColorName (comma delim.) string) or #{hn}hnhnhn (hex. string)	
<b>_MessageSize</b>	numeric	Textsize of the message. Default is 12.
<b>_MessageColor</b>	{Alpha,}Red,Green,Blue (comma delimited string) or _{Alpha,}ColorName (comma delim.) string) or #{hn}hnhnhn (hex. string)	Color of the message.
<b>_Button1Size</b>	numeric	

Table of options		
Key	Value	Description
<b>_Button1Color</b>	{Alpha,}Red,Green,Blue (comma delimited string) or _{Alpha,}ColorName ({comma delim.} string) or #{hn}hnhnhn (hex. string)	
<b>_Button2Size</b>	numeric	
<b>_Button2Color</b>	{Alpha,}Red,Green,Blue (comma delimited string) or _{Alpha,}ColorName ({comma delim.} string) or #{hn}hnhnhn (hex. string)	
<b>_Button3Size</b>	numeric	
<b>_Button3Color</b>	{Alpha,}Red,Green,Blue (comma delimited string) or _{Alpha,}ColorName ({comma delim.} string) or #{hn}hnhnhn (hex. string)	
<b>_Cancelable</b>	numeric	If > 0 the dialog is cancel-able. Default is 1.

**Dialog.multi** <retBut\_nval>, <retChk\_Array[]>, <items\_Array\$[]>, <title\_sexp>{**{}{}**, <button1\_sexp>}, <button2\_sexp>}, <button3\_sexp>}, <bndPtr\_nexp>}

Starts a multi-selection dialog. The keystrokes are returned by <retBut\_nval>. If <retBut\_nval> is assigned a negative number before the command is called, a timer is set and this counts down in milliseconds before it ends the dialog.

The results will be returned by the array <retChk\_Array[]>. This array has to be predefined with given checks.

The items of the list are given by <items\_Array\$[]>.

The length of both arrays have to be equal.

The buttons will be optionally set by <button1\_sexp> (positive), <button2\_sexp> (neutral) and <button3\_sexp> (negative).

More layout specifications can be set by the bundle <bndPtr\_nexp>. See the tables of Dialog.cust.call for more.

Example:

```
ARRAY.LOAD items$, "Kurt","Franz", "Maria","Luca", "Marc", "Nicolas","Tanya","Bob"
ARRAY.LOAD retChk[], 0,1,0,1,0,1,0,1
DIALOG.MULTI retBut , retChk[], items$, "Title", "OK", "Cancel"
PRINT retBut
PRINT "Checked: ", retChk[1], retChk[2], retChk[3], retChk[4], retChk[5], retChk[6];
PRINT retChk[7], retChk[8]
```

**Dialog.single** <retBut\_nval>, <retSel\_nvar>, <items\_Array\$[]>, <title\_sexp>{**{}{}**, <button1\_sexp>}, <button2\_sexp>}, <button3\_sexp>}, <bndPtr\_nexp>}

Starts a single-selection dialog. The keystroke is returned by <retBut\_nval>. If <retBut\_nval> is assigned a negative number before the command is called, a timer is set and this counts down in milliseconds before it ends the dialog.

The result will be returned by <retSel\_nvar>. This variable can be predefined with a given value.

The items of the list are given by <items\_Array\$[]>.

The buttons will be optionally set by <button1\_sexp> (positive), <button2\_sexp> (neutral) and <button3\_sexp> (negative).

More layout specifications can be set by the bundle <bndPtr\_nexp>. See the tables of Dialog.cust.call for more.

Example:

```
ARRAY.LOAD stdColors$, "_Black","_White","_Gray","_Red","_Green","_Blue","_Cyan"
retSel = 4
DIALOG.SINGLE retBut, retSel, stdColors$, "Choose a Color", "OK"
PRINT stdColors$[retSel]
```

**Dialog.message** {<title\_sexp>}, {<message\_sexp>}, <sel\_nvar> {**{}{}**, <button1\_sexp>}, <button2\_sexp>}, <button3\_sexp>}, <layout\_bundle\_nexp>}

Generates a dialog box with a title, a message, and up to three buttons. When the user taps a button, the number of the selected button is returned in <sel\_nvar>. If the user taps the screen outside of the message dialog or presses the BACK key, then the returned value is 0.



The string <title\_sexp> becomes the title of the dialog box. The string <message\_sexp> is displayed in the body of the dialog, above the buttons. The strings <button1\_sexp> ([positive](#)), <button2\_sexp> ([neutral](#)), and <button3\_sexp> ([negative](#)) provide the labels on the buttons.

You may have 0, 1, 2, or 3 buttons. On most devices, the buttons are numbered from right-to-left, because Android style guides recommend the positive action on the right and the negative action on the left. Some devices differ. On compliant devices, tapping the right-most button returns 1.

All of the parameters except the selection index variable <sel\_nvar> are optional. If any parameter is omitted, the corresponding part of the message dialog is not displayed. Use commas to indicate omitted parameters (see Optional Parameters).

Examples:

```
Dialog.Message "Hey, you!", "Is this ok?", ok, "Sure thing!", "Don't care", "No way!"
Dialog.Message "Continue?", , go, "YES", "NO"
Dialog.Message , "Continue?", go, "YES", "NO"
Dialog.Message , , b
```

The first command displays a full dialog with a title, a message, and three buttons.

The second command displays a box with a title and two buttons – note that the **YES** button will be on the right and the **NO** button on the left. The third displays the same information, but it looks a little different because the text is displayed as the message and not as the title. Note the commas.

The fourth command displays nothing at all. The screen dims and your program waits for a tap or the BACK key with no feedback to tell the user what to do.

[Is a negative <sel\\_nvar> at command start given, the message dialog will be finished in <sel\\_nvar> \\* -1 milliseconds. In this case <sel\\_nvar> returns 0.0.](#)

[Note, that in some Android versions in Graphic Mode the width of the box is shrinking according to the text size. Some spaces at the end of the title or the message will help if needed.](#)

The <title\_sexp> and <message\_sexp> support also HTML code. So a line break needs a "<br>" instead of "\n". [See the SELECT layout description table under \\_TextHtml.](#) Some characters have to be changed "<" into "&lt;" and ">" into "&gt;" as an example.

The optional layout bundle <layout\_bundle\_nexp> controls the Dialog.message output layout:

Table of layout control options		
Key	Value	Description
<b>_Style</b>	<b>_Default</b>	Default theme
	<b>_Dark</b>	Dark theme and Autosize mode
	<b>_Bright</b>	Bright theme and Autosize mode
<b>_Icon</b>	String	File path of a header icon
<b>_PositionH</b>	<b>_Left</b>	
	<b>_Center</b>	Default
	<b>_Right</b>	
<b>_PositionV</b>	<b>_Top</b>	
	<b>_Center</b>	Default
	<b>_Bottom</b>	
<b>_MessageSize</b>	numeric	

Table of layout control options		
Key	Value	Description
<b>_MessageColor</b>	{Alpha,}Red,Green,Blue (comma delimited string) or _{Alpha,}ColorName ({comma delim.} string) or #{hn}hnhnhn (hex. string)	
<b>_Button1Size</b>	numeric	
<b>_Button1Color</b>	{Alpha,}Red,Green,Blue (comma delimited string) or _{Alpha,}ColorName ({comma delim.} string) or #{hn}hnhnhn (hex. string)	
<b>_Button2Size</b>	numeric	
<b>_Button2Color</b>	{Alpha,}Red,Green,Blue (comma delimited string) or _{Alpha,}ColorName ({comma delim.} string) or #{hn}hnhnhn (hex. string)	
<b>_Button3Size</b>	numeric	
<b>_Button3Color</b>	{Alpha,}Red,Green,Blue (comma delimited string) or _{Alpha,}ColorName ({comma delim.} string) or #{hn}hnhnhn (hex. string)	
<b>_Cancelable</b>	numeric	If > 0 the dialog is cancel-able. Default is 1.

**Dialog.select** <sel\_nvar>, <Array\$[]>|<list\_nexp> {{, <title\_sexp>}, <layout\_bundle\_nexp>}

Generates a dialog box with a list of choices for the user. When the user taps a list item, the index of the selected line is returned in the <sel\_nvar>. If the user taps the screen outside of the selection dialog or presses the BACK key, then the returned value is 0. <Array\$[]> is a string array that holds the list of items to be selected. The array is specified without an index but must have been previously dimensioned or loaded via Array.load. As an alternative to an array, a string-type list may be specified in the <list\_nexp>. The <title\_sexp> is an optional string expression that will be displayed at the top of the selection dialog. If the parameter is not present, or the expression evaluates to an empty string (""), the dialog box will be displayed with no title.

The `<title_sexp>` supports also HTML code. See the SELECT layout description table under `_TextHtml`.

The optional layout bundle <layout\_bundle\_nexp> controls the Dialog.select output layout:

Table of layout control options		
Key	Value	Description
<b>_Style</b>	<b>_Default</b>	Default theme
	<b>_Dark</b>	Dark theme and Autosize mode
	<b>_Bright</b>	Bright theme and Autosize mode
<b>_Icon</b>	String	File path of a header icon
<b>_PositionH</b>	<b>_Left</b>	
	<b>_Center</b>	Default
	<b>_Right</b>	
<b>_PositionV</b>	<b>_Top</b>	
	<b>_Center</b>	Default
	<b>_Bottom</b>	
<b>_Button1</b>	Name of Button 1	<b>Positive</b> button Returns -1
<b>_Button1Size</b>	numeric	
<b>_Button1Color</b>	{Alpha,}Red,Green,Blue (comma delimited string) or _{Alpha,}ColorName ({comma delim.} string) or #{hn}hnhnhn (hex. string)	
<b>_Button2</b>	Name of Button 2	<b>Neutral</b> button Returns -2
<b>_Button2Size</b>	numeric	
<b>_Button2Color</b>	{Alpha,}Red,Green,Blue (comma delimited string) or _{Alpha,}ColorName ({comma delim.} string) or #{hn}hnhnhn (hex. string)	
<b>_Button3</b>	Name of Button 3	<b>Negative</b> button Returns -3
<b>_Button3Size</b>	numeric	
<b>_Button3Color</b>	{Alpha,}Red,Green,Blue (comma delimited string) or _{Alpha,}ColorName ({comma delim.} string) or #{hn}hnhnhn (hex. string)	
<b>_Cancelable</b>	numeric	If > 0 the dialog is cancel-able. Default is 1.

**Select** <sel\_nvar>, <Array\$[]>|<list\_nexp>, <title\_sexp>{{{, <message\_sexp>}, <press\_nvar> }, <layout\_bundle\_nexp>}

The Select command generates a new screen with a list of choices for the user. When the user taps a screen line, the index of the selected line is returned in the <sel\_nvar>. If the user presses the BACK key, then the returned value is 0.

<Array\$[]> is a string array that holds the list of items to be selected. The array is specified without an index but must have been previously dimensioned, loaded via Array.load, or created by another command.

As an alternative to an array, a string-type list may be specified in the <list\_nexp>.

The <title\_sexp> is an ~~optional~~ string expression that is placed into the title bar at the top of the selection screen. ~~If the parameter is not present, the screen displays a default title.~~ If the expression evaluates to an empty string (""), the title is blank.

The <message\_sexp> is an optional string expression that is displayed in a short Popup message. If the message is an empty string (""), there is no Popup. If the parameter is absent, the <title\_sexp> string is used instead, but if the <title\_sexp> is also missing or empty, there is no Popup.

If the optional <press\_nvar> is present, the type of user touch a short tap (0), a long press (1) ~~or a double tap (2)~~ is returned in the <press\_nvar>. ~~The delay for detecting the double tap corresponds to the default Android system settings. Its value will be 0 (false) if the touch was a short tap. Its value will be 1 (true) if the touch was a long press.~~

Use commas to indicate omitted optional parameters (see Optional Parameters).

Is a negative <sel\_nvar> at command start given, the Select dialog will be finished in <sel\_nvar> \* -1 milliseconds. In this case <sel\_nvar> returns 0.0.

The optional layout bundle <layout\_bundle\_nexp> controls the ~~Select output layout:~~

Table of layout control options		
Key	Value	Description
<b>_ textSize</b>	numeric	
<b>_ TextColor</b>	{Alpha,}Red,Green,Blue (comma delimited string) or _{Alpha,}ColorName ({comma delim.} string) or #{hn}hnhnhn (hex. string)	Note, _ TextFont or _ TextStyle is needed also!
<b>_ TextBackgroundColor</b>	{Alpha,}Red,Green,Blue (comma delimited string) or _{Alpha,}ColorName ({comma delim.} string) or #{hn}hnhnhn (hex. string)	Has to be "0,0,0,0" if you want a background color, wallpaper or bitmap Note, _ TextFont or _ TextStyle is needed also!
<b>_ TextFont</b>	_ Default	
	_ Serif	
	_ Sans_Serif	
	_ Monospace	
<b>_ TextStyle</b>	_ Normal	


Table of layout control options		
Key	Value	Description
	<code>_Bold</code>	
	<code>_Bold_Italic</code>	
	<code>_Italic</code>	
<code>_TextHtml</code>	0 or 1 (numeric)	<p>Returns displayable styled text from the provided HTML string. But not all tags are supported. Any <code>&lt;img&gt;</code> tags in the HTML will display an image. Absolute ("<a href="#">file:///</a>") and relative paths are allowed. The image size has to be scaled before, because <code>h=</code> and <code>w=</code> are ignored. See <code>_HtmlBitmapScale</code>.</p>  <p>Uses parts of TagSoup library to handle real HTML, including all of the brokenness found in the wild.</p> <p> <code>&lt;a href="..."&gt;</code>  <code>&lt;b&gt;</code>  <code>&lt;big&gt;?</code>  <code>&lt;blockquote&gt;</code>  <code>&lt;br&gt;</code>  <code>&lt;del&gt;</code>  <code>&lt;cite&gt;</code>  <code>&lt;dfn&gt;</code>  <code>&lt;div align="..."&gt;?</code> Use instead <code>chr\$(1564)</code> [Arabic Letter] at line begin for <code>align='right'</code>  <code>&lt;em&gt;</code>  <code>&lt;font size="..." color="..." face="..."&gt;</code>  <code>&lt;h1&gt;</code>, <code>&lt;h2&gt;</code>, <code>&lt;h3&gt;</code>, <code>&lt;h4&gt;</code>, <code>&lt;h5&gt;</code>, <code>&lt;h6&gt;</code>  <code>&lt;i&gt;</code>  <code>&lt;img src="..."&gt;</code>  <code>&lt;p&gt;</code>  <code>&lt;small&gt;</code>  <code>&lt;strike&gt;?</code> <code>&lt; A.7</code>  <code>&lt;strong&gt;</code>  <code>&lt;sub&gt;</code>  <code>&lt;sup&gt;</code>  <code>&lt;tt&gt;?</code>  <code>&lt;u&gt;</code> </p> <p><b>Replace Space with <code>&amp;#160;</code>, &amp; with <code>&amp;amp;</code>, &lt; with <code>&amp;lt;</code>, &gt; with <code>&amp;gt;</code>, " with <code>&amp;quot;</code>; if necessary.</b></p>

Table of layout control options		
Key	Value	Description
<b>_HtmlTextSelectable</b>	0 or 1 (numeric)	Does only work in conjunction with _TextHtml, but the item selection works only with a <b>long</b> click.
<b>_HtmlBitmapScale</b>	-1, 0, > 0 (numeric)	Does only work in conjunction with _TextHtml. Scales the included bitmaps in the following ways: -1 No scaling, 0 (default) Only scaling proportional to the screen resolution > 0 Proportional to the font size If it is 1 the bitmap height is the same as the font size.
<b>_DividerColor</b>	{Alpha,}Red,Green,Blue (comma delimited string) or _{Alpha,}ColorName ({comma delim.} string) or #{hn}hnhnhn (hex. string)	
<b>_DividerFilename</b>	bitmap file path	
<b>_DividerHeight</b>	numeric	
<b>_BackgroundWallpaper</b>	0 or 1 (numeric)	Min. Jelly Bean 4.1 (API 16)
<b>_BackgroundColor</b>	{Alpha,}Red,Green,Blue (comma delimited string) or _{Alpha,}ColorName ({comma delim.} string) or #{hn}hnhnhn (hex. string)	
<b>_BackgroundFilename</b>	bitmap file path	Min. Jelly Bean 4.1 (API 16)
<b>_Orientation</b>	(numeric)  -1, 0, 1, 2 or 3	The value sets the orientation of screen as follows: -1 = Orientation depends upon the sensors. 0 = Orientation is forced to Landscape. 1 = Orientation is forced to Portrait. 2 = Orientation is forced to Reverse Landscape. 3 = Orientation is forced to Reverse Portrait.
<b>_SetSelection</b>	(numeric)	Sets a pre selected item. The item will not be selected but it will still be positioned appropriately. If the specified selection position is less than 1, then the item at position 1 will be selected.

Table of layout control options		
Key	Value	Description
<b>_StackFromBottom</b>	0 or 1 (numeric)	If 1 pin the view's content to the bottom edge, 0 to pin the view's content to the top edge
<b>_Subtitle</b>	String	Set the Action bar's subtitle.
<b>_TitleShow</b>	0 or 1 (numeric)	If 1 (default) Show the Action bar if it is not currently showing. It will resize application content to fit the new space available. If 0 Hide the Action bar if it is currently showing. It will resize application content to fit the new space available.
<b>_TitleIcon</b>	Icon file path	Add a large icon to the notification content view. <a href="http://romannurik.github.io/AndroidAssetStudio/index.html">http://romannurik.github.io/AndroidAssetStudio/index.html</a>
<b>_TitleHomeEnabled</b>	0 or 1 (numeric)	Set whether to include the application home accordance in the Action bar. Home is presented as an activity icon. Have to be 1 if you want to show the icon. Have to be 0 if you want to hide the icon. The default setting is API dependent.
<b>_TitleBackground</b>	Background file path	




Table of layout control options		
Key	Value	Description
<b>_TitleHtml</b>	0 or 1 (numeric)	<p>Returns displayable styled text from the provided HTML string. But not all tags are supported.</p>  <p>Uses parts of TagSoup library to handle real HTML, including all of the brokenness found in the wild.</p> <p>&lt;b&gt; &lt;big&gt; &lt;font size="..." color="..." face="..."&gt; &lt;h1&gt;, &lt;h2&gt;, &lt;h3&gt;, &lt;h4&gt;, &lt;h5&gt;, &lt;h6&gt; &lt;i&gt; &lt;small&gt; &lt;strike&gt;? &lt; A.7 &lt;strong&gt; &lt;sub&gt; &lt;sup&gt; &lt;tt&gt;? &lt;u&gt;</p> <p><b>Replace Space with &amp;#160, &amp; with &amp;amp, &lt; with &amp;lt, &gt; with &amp;gt, " with &amp;quot; if necessary.</b></p> <p>Usable for Title and Subtitle. Keep in mind that the ActionBar height will not be expanded.</p>
<b>_ShowStatusbar</b>	0, 1 or 2 (numeric)	<p>If 1 (default) The Status bar will be displayed.</p> <p>If 2 The Status bar will be transparent displayed. Min. Lollipop 5.0 (API 21)</p> <p>If 0 The Status bar will be hidden to the background. Min. Nougat 7.0 (API 24) Will be switched to option 2 or 1 if the current API level is lower.</p>
<b>_StatusbarColor</b>	{Alpha,}Red,Green,Blue (comma delimited string) or _{Alpha,}ColorName ({comma delim.} string) or #{hn}hnhnhn (hex. string)	Min. Lollipop 5.0 (API 21)

Table of layout control options		
Key	Value	Description
<b>_StatusbarLight</b>	0 or 1 (numeric)	If 0 (default) The Status bar background is dark. In this case the <b>bar content</b> will be <b>light</b> . If 1 The Status bar background is light. In this case the <b>bar content</b> will be <b>dark</b> . Min. Lollipop 5.0 (API 21)
<b>_ShowNavigationbar</b>	0, 1 or 2 (numeric)	If 1 (default) The Navigation bar will be displayed. If 2 The Navigation bar will be transparent displayed. Min. Lollipop 5.0 (API 21) If 0 The Navigation bar will be hidden to the background. Min. Nougat 7.0 (API 24) Will be switched to option 2 or 1 if the current API level is lower.
<b>_NavigationbarColor</b>	{Alpha,}Red,Green,Blue (comma delimited string) or _{Alpha,}ColorName ({{comma delim.} string) or #{hn}hnhnhn (hex. string)	Min. Lollipop 5.0 (API 21)
<b>_NavigationbarLight</b>	0 or 1 (numeric)	If 0 (default) The Navigation bar background is dark. In this case the <b>bar content</b> will be <b>light</b> . If 1 The Navigation bar background is light. In this case the <b>bar content</b> will be <b>dark</b> . Min. Lollipop 5.0 (API 21)

Note, that you should scale the text size and the divider height in conjunction to the detected screen size.

If you want to change something in the layout bundle, it is sufficient to make the changes only in the bundle. Note that every change affects the whole thing.

See also `Console.layout`, `Console.title`

Example:

```
ARRAY.LOAD months$[], "January", "February"~
"March", "April", "May", "June", "July", "August"~
"September", "October", "November", "December"
```

```
bundle.put layout,"_TextSize", 30
bundle.put layout,"_TextColor", "0,0,255"
bundle.put layout,"_TextBackgroundColor", "250,250,250,250"
bundle.put layout,"_TextStyle", "_Bold"
bundle.put layout,"_SetSelection", 10
! bundle.put layout,"_StackFromBottom", 1
! bundle.put layout,"_DividerFilename", "cartman.png"
bundle.put layout,"_DividerColor", "255,0,255,0"
bundle.put layout,"_DividerHeight", 5
! bundle.put layout,"_BackgroundFilename", "cartman.png"
tLong = 10
SELECT month, months$[], msg$, tLong ,layout
PRINT months$[month]
```

**Text.input <svar>{{{, { <text\_sexp> } , <title\_sexp>}, <suggestionOff\_nexp>}, <layout\_bundle\_nexp>}**

This command is similar to "Input" except that it is used to input and/or edit a large quantity of text. It opens a new window with scroll bars and full text editing capabilities. You may set the title of the new window with the optional <title\_sexp> parameter.

If the optional <text\_sexp> is present then that text is loaded into the text input window for editing. If <text\_sexp> is not present then the text.input text area will be empty. If <title\_sexp> is needed but text.input text area is to be initially empty, use two commas to indicate the <sexp> specifies the title and not the initial text.

When done editing, tap the Finish **button item in the menu**. The edited text is returned in <svar>.


If you tap the BACK key **or the Stop item in the menu** then all text editing is discarded. <svar> returns the original <sexp> text.

The optional parameter <suggestionOff\_nexp> sets an input type. If it 1, text suggestion is switched off. Default is 0, text suggestion is switched on.

The following example grabs the Sample Program file, **f01\_commands.bas**, to string s\$. It then sends s\$ to text.input for editing. The result of the edit is returned in string r\$. r\$ is then printed to console.

```
GRABFILE s$, "../source/ Sample_Programs/f01_commands.bas"
TEXT.INPUT r$,s$
PRINT r$
END
```

The optional layout bundle <layout\_bundle\_nexp> controls the Text.input output layout:

Table of layout control options		
Key	Value	Description
<b>_TextHtml</b>	0 or 1 (numeric)	<p>Returns displayable styled text from the provided HTML string. But not all tags are supported. Any &lt;img&gt; tags in the HTML will display an image. Absolute ("<a href="#">file:///</a>") and relative paths are allowed. The image size has to be scaled before, because h= and w= are ignored. See <a href="#">_HtmlBitmapScale</a>.</p>  <p>Uses parts of TagSoup library to handle real HTML, including all of the brokenness found in the wild.</p> <p>&lt;a href="..."&gt;          &lt;b&gt;          &lt;big&gt;?          &lt;blockquote&gt;          &lt;br&gt;          &lt;del&gt;          &lt;cite&gt;          &lt;dfn&gt;          &lt;div align="..."&gt;? Use instead chr\$(1564) [Arabic Letter] at line begin for align=' right'          &lt;em&gt;          &lt;font size="..." color="..." face="..."&gt;          &lt;h1&gt;, &lt;h2&gt;, &lt;h3&gt;, &lt;h4&gt;, &lt;h5&gt;,          &lt;h6&gt;          &lt;i&gt;          &lt;img src="..."&gt;          &lt;p&gt;          &lt;small&gt;          &lt;strike&gt;? &lt; A.7          &lt;strong&gt;          &lt;sub&gt;          &lt;sup&gt;          &lt;tt&gt;?          &lt;u&gt;</p> <p><b>Replace          Space with &amp;#160;,          &amp; with &amp;amp;,          &lt; with &amp;lt;,          &gt; with &amp;gt;,          " with &amp;quot;,          if necessary.</b></p>

**Popup <message\_sexp> {{, <x\_nexp>}{, <y\_nexp>}{, <duration\_lexp>}}**

Pops up a small message for a limited duration. The message is <message\_sexp>.

All of the parameters except the message are optional. If omitted, their default values are 0. Use commas to indicate omitted parameters (see Optional Parameters).

The simplest form of the **Popup** command, **Popup "Hello!"**, displays the message in the center of the screen for two seconds.

The default location for the Popup is the center of the screen. The optional <x\_nexp> and <y\_nexp> parameters give a displacement from the center. The values may be negative.

Select the duration of the Popup, either 2 seconds or 4 seconds, with the optional <duration\_lexp> "long flag". If the flag is false (the expression evaluates to 0) the message is visible for 2 seconds. If the flag is true (non-zero) the message is visible for 4 seconds. If the flag is omitted the duration is short.

The popup will be created outside from the current app in an Android message queue.

These popup messages will be shown in any case step by step, also if the program is finished.

## Clipboard

Unless your app is the default input method editor (IME) or is the app that currently has focus, your app cannot access clipboard data on Android 10 or higher.

### Clipboard.info <bundle\_nvar>

Describes the current contents of the clipboard into the Bundle <bundle\_nexp>.

Table of description arguments		
Key	Value	Description
<b>_Label</b>	String	The label of the clipboard content. Often also null is returned.
<b>_Timestamp</b>	Numeric	Timestamp in milliseconds since 1970-01-01T00:00:00. Android 8.0+ needed.
<b>_MimeType_1</b>	String	Mime type 1, if available
<b>_MimeType_2</b>	String	Mime type 2, if available

### Clipboard.get <svar>{, <type\_sexp>}

Copies the current content of the clipboard into <svar>. The optional <type\_sexp> specifies the type of content. Available are "\_Text" (default), "\_Html", "\_ToHtml" and "\_Styled". In case of "\_Styled" any text that would be returned as HTML formatting will be returned as text with Android style spans. To convert the main content into HTML text use "\_ToHtml".

### Clipboard.put <sexp>{{{, <label\_sexp>}, <type\_sexp>}, <html\_sexp>}

Places <sexp> into the clipboard. An optional label can be set with the <label\_sexp>. A description of the predefined content type, such as "text", "html", is recommended, but you can also indicate the origin of the use. The optional <type\_sexp> defines the content type like "\_Text" (default), "\_Html" and "\_Uri".

Example:

```
CLIPBOARD.PUT "Text without html", "two times text", "_Html", "Text with html"
CLIPBOARD.INFO bPointer
DEBUG.ON
DEBUG.DUMP.BUNDLE bPointer
CLIPBOARD.GET c$, "_Text"
PRINT c$
CLIPBOARD.GET c$, "_Html"
PRINT c$
```

## Scheduler Interrupt and Commands

You can set a scheduler that will interrupt the execution of your program at some set time or interval. When the scheduler expires, BASIC! jumps to the statements following the **OnSched:** label. When you have done whatever you need to do to handle this scheduler event, you use the **Sched.resume** command to resume execution of the program at the point where the scheduler interrupt occurred.

The scheduler cannot interrupt an executing command. When the scheduler expires, the current command is allowed to complete. Then the scheduler interrupt code after the **OnSched:** label executes. If the current command takes a long time to finish, it may appear that your scheduler is late.

The scheduler cannot interrupt another interrupt. If the scheduler expires while any interrupt event handler is running, the **OnSched:** interrupt will be delayed. If the scheduler expires while the **OnSched:** interrupt handler is running, the scheduler event will be lost. The **OnSched:** interrupt code must exit by running **Sched.resume**, or the scheduler interrupt can occur only once.

### **Sched.set** <firstInterrupt\_nexp>, <interval\_nexp> {,<date\_flag\_nexp>}

Sets a scheduler that will interrupt program execution after the in <firstInterrupt\_nexp> specified time or date in milliseconds. The parameter <interval\_nexp> sets a repeatedly program execution interrupt with the specified time interval in milliseconds. The optionally flag <date\_flag\_nexp> defines whether <firstInterrupt\_nexp> is a date driven time (>0) or a time distance (≤0). The date driven time is default. If <firstInterrupt\_nexp> and <interval\_nexp> are equal and <date\_flag\_nexp> is 0 the functionality is the same as **Timer.set**. That is a good option, if you need **two timers**. The program must also contain an **OnSched:** label.

See also **Timer.set**.

```
t = TIME(2018, 1, 1, 0, 0, 1)
```

```
SCHED.SET t, 0 % Happy New Year 2018! One time interrupt.
```

! Is your device switched off at this date, the interrupt occur at the next program start.

```
t = TIME() + 4000
```

```
SCHED.SET t, 2000, 1
```

! First Interrupt at current time + 4 seconds and the interrupt repeats every 2 seconds.

```
SCHED.SET 0, 2000, 0 % Interrupt immediately and repeats every 2 seconds.
```

```
SCHED.SET 2000, 2000, 0 % It is the same as TIMER.SET 2000
```

```
SCHED.SET 2000, 0, 0 % Interrupt one time in 2 seconds.
```

### **OnSched:**

Interrupt label for the scheduler interrupt. (See "Interrupt Labels".)

### **Sched.resume**

Resumes execution at the point in the BASIC! program where the **OnSCHED:** interrupt occurred.



## Sched.clear

Clears the repeating scheduler. No further scheduler interrupts will occur.

## Sensor Commands

### Sensors.list <sensor\_array\$[]>{, <all\_nexp>}

Writes information about the sensors available on the Android device into the <sensor\_array\$[]> parameter. If the array exists, it is overwritten. Otherwise a new array is created. The result is always a one-dimensional array.

The array elements contain the names and types of the available sensors. For example, one element may be "Gyroscope, Type = 4".

Is <all\_nexp> = 1 and Android 5+ all available information will be presented. Default <all\_nexp> is set to 0.

The following program snippet prints the elements of the sensor list.

```
SENSORS.LIST sensorarray$[]  
ARRAY.LENGTH size, sensorarray$[]  
FOR index = 1 TO size  
    PRINT sensorarray$[ index ]  
NEXT index  
END
```

### Sensors.read <sensor\_type\_nexp>, <p1\_nvar>, <p2\_nvar>, <p3\_nvar> {, <param\_array[]>}

This command returns that latest values from the sensors specified by the "sensor\_type" parameters. The values are returned are placed into the p1, p2 and p3 parameters. The meaning of these parameters depends upon the sensor being read. Not all sensors return all three parameter values. In those cases, the unused parameter values will be set to zero. See [Android's Sensor Event](#) web page for the meaning of these parameters.

The optional <param\_array[]> returns an array with three or more parameters.

Sensortypes up to 30 and parameters up to 20 are supported.

## Special Floating Point Commands

### Is\_NaN(<nexp>)

Returns 1.0 (true) if <nexp> is a NaN (Not a Number) Floating Point number else 0.0 if false.

### Is\_Infinite(<nexp>)

Returns 1.0 (true) if <nexp> is an Infinity Floating Point number else 0.0 if false.

Note, that Is\_Infinite ends with an **e**!

## Graphical Commands

**Within(<sexp>, <arg1>, <arg2>, <arg3>, <arg4>...)**

Returns <> 0 (true) if the search for a graphical object described by <sexp> is successful. In other cases 0.0 false is returned.

As an example read for "\_point\_polygon": **"With point in polygon?"**.

If the point is directly onto a line, it is true also.

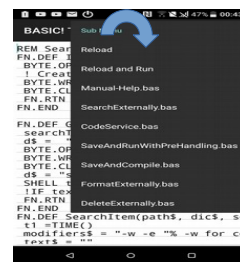
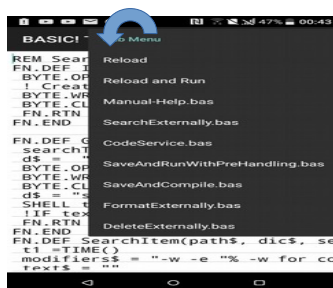
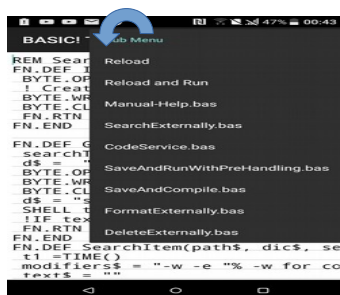
Is a positive number returned, the polygon was drawn clockwise.

Is a negative number returned, the polygon was drawn counterclockwise.

If +1 or -1 is returned, the point is inside the polygon.

If +2 or 2 is returned, the point is inside an inner polygon of the polygon.

<sexp>/description	Argument number	Value type	Value	Default
<b>"_point_polygon"</b>				
Point x	1	Double	Double	
Point y	2	Double	Double	
List of Polygon points as x values or x, y pairs	3	List numeric	Double	
{List of Polygon points as y values} (Option)	4	List numeric	Double	
For more information see <a href="http://geomalgorithms.com/a03-_inclusion.html">http://geomalgorithms.com/a03-_inclusion.html</a>				



Read for "\_polygon\_polygon": **"With polygon in polygon?"**.

If at least one point of the first polygon is inside or is directly onto a line of the second polygon, it returns true (1).

<sexp>/description	Argument number	Value type	Value	Default
<b>"_polygon_polygon"</b>				
List of Polygon 1 points as <del>x-values</del> or x, y pairs	1	List numeric	Double	
List of Polygon 2 points as <del>x-values</del> or x, y pairs	2	List numeric	Double	

## Hide and Show Commands

### **Gr.hide** <object\_number\_nexp>{, <object\_number\_nexp> ...}

Hides the objects with the specified Object Numbers. If the Object is a Group, all of the Graphical Objects in the Group are hidden. This change will not be visible until the **Gr.render** command is called.

### **Gr.show** <object\_number\_nexp>{, <object\_number\_nexp> ...}

Shows (unhides) the objects with the specified Object Numbers. If the Object is a Group, all of the Graphical Objects in the Group are shown. This change will not be visible until the **Gr.render** command is called.

### **Gr.toFront** <object\_number\_nexp>{, <object\_number\_nexp> ...}

Moves the object with the specified Object Number visually to the Front. That means this element is the last one on the Display List. If more than one object is specified, this command works in the order of the specified objects. This change will not be visible until the **Gr.render** command is called. Note, this command works only in conjunction with shown single objects.

### **Gr.behind** <object\_behind\_nexp>, <object\_number\_nexp>

Moves the object with the specified Object Number <object\_behind\_nexp> in behind the object <object\_number\_nexp> visually. That means the element <object\_behind\_nexp> is in front of the one specified by <object\_number\_nexp> on the Display List. This change will not be visible until the **Gr.render** command is called. Note, this command works only in conjunction with shown single objects.

### **Gr.inFront** <object\_inFront\_nexp>, <of\_object\_nexp>

Moves the object with the specified Object Number <object\_inFront\_nexp> in front of the object <of\_object\_nexp> visually. That means the element <object\_inFront\_nexp> is behind the one specified by <of\_object\_nexp> on the Display List. This change will not be visible until the **Gr.render** command is called. Note, this command works only in conjunction with shown single objects.

### **Gr.toBack** <object\_number\_nexp>{, <object\_number\_nexp> ...}

Moves the object with the specified Object Number visually to the Back. That means this element is the first one on the Display List. If more than one object is specified, this command works in the order of the specified objects. This change will not be visible until the **Gr.render** command is called. Note, this command works only in conjunction with shown single objects.

### **Gr.render**

Sets the graphic screen to invalidate. The system will schedule a redraw of This command displays all objects that are listed in the current working Display List and are not marked as hidden. It is not necessary to have a **Pause** command after a **Gr.render**. The **Gr.render** command will not complete until the contents of the Display List have been fully displayed. **Gr.render** always waits until the next screen refresh. Most Android devices refresh the screen 60 times per second; your device may be faster or slower. Therefore, if you execute two consecutive **Gr.render** commands, there will be a delay of 16.7 milliseconds (on most devices) between the two commands.

For smooth animation, try to avoid doing more than 16.7 ms of work between **Gr.render** commands, to achieve the maximum refresh rate. This is not a lot of time for a BASIC! program, so you may have to settle for a lower frame rate. However, there is no benefit to trying to render more often than 16.7 ms.

If BASIC! is running in the background (see **Background()** function and **Home** command), **Gr.render** will not execute. It will pause your program until you return BASIC! to the foreground.

Keep in mind, that under some circumstances (software rendering) a graphic object can be displayed before **Gr.render** is called. Hardware rendering is the default option and can only be changed by `GR.set.acceleration`.

If the program has to do a lot of work in the background before the result should be displayed, do not draw or change any objects until the background work is complete.

See also `GR.set.acceleration`

**Notify <title\_sexp>, <subtitle\_sexp>, <alert\_sexp>, <wait\_lexp>{{{, <options\_bundle\_nexp>}, <notification\_id\_nvar>}, <notified\_id\_nvar>}**

This command will cause a Notify object to be placed in the Notify (Status) bar. The Notify object displays the BASIC! app icon and the <alert\_sexp> text. The user taps the Notify object to open the notification window. Your program's notification displays the <title\_sexp> and <subtitle\_sexp> text.

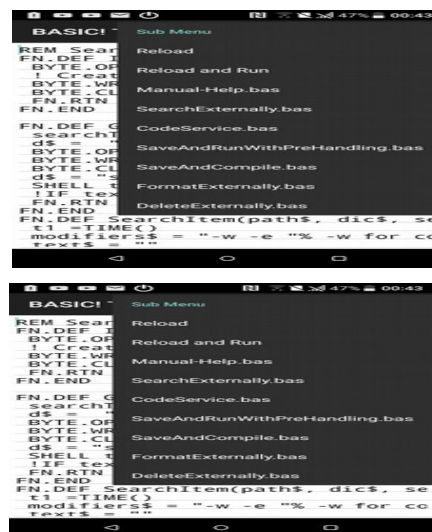
The code snippet and screenshots shown below demonstrate the placement of the parameter strings.

If <wait\_lexp> is not zero (true), then the execution of the BASIC! program will be suspended until the user taps the Notify object. The optional numeric variable <notified\_id\_nvar> returns in this case the notification id of the touched notification. If <notified\_id\_nvar> returns -1, no notification touch was identified. If the value <wait\_lexp> is zero (false), the BASIC! program will continue executing.

The Notify object will be removed when the user taps or slides the object, or when the program exits. See also the bundle keys \_AutoCancel and \_Ongoing later.

Example:

```
Print "Executing Notify"
Notify "BASIC! Notify", "Tap to resume running program",~
"BASIC! Notify Alert", 1
! Execution is suspended and waiting for user to tap the Notify Object
Print "Notified"
```



The optional options bundle <options\_bundle\_nexp> controls the notification:

Key	Value	
<b>_SmallIcon</b>	Small icon file path	Sets the small icon, which will be used to represent the notification in the status bar. The platform template for the expanded view will draw this icon normally in the left. It is a special bitmap. The visible content is defined only by the alpha channel. <a href="http://romannurik.github.io/AndroidAssetStudio/icons-notification.html">http://romannurik.github.io/AndroidAssetStudio/icons-notification.html</a>
<b>_LargeIcon</b>	Large icon file path	Add a large icon to the notification content view. <a href="http://romannurik.github.io/AndroidAssetStudio/index.html">http://romannurik.github.io/AndroidAssetStudio/index.html</a>
<b>_Color</b>	{Alpha,}Red,Green,Blue (comma delimited string) or _{Alpha,}ColorName ({comma delim.} string) or #{hn}hnhnhn (hex. string)	Min. Lollipop 5.0 (API 21)
<b>_Progress</b>	Max, progress, animated	Progress bar, Max → maximal (numeric value), progress → between 0 and maximal (numeric value), animated > 0 → true
<b>_Sound</b>	sound file path	Deprecated but Note: Beginning with Android 8.1 (API level 27), apps cannot make a notification sound more than once per second. If your app posts multiple notifications in one second, they all appear as expected, but only the first notification per second makes a sound. Beginning with Android 9 (API level 28) system sounds take over.
<b>_AutoCancel</b>	0 or > 0 (numeric)	Makes this notification automatically dismissed when the user touches or slides it. Default is true (> 0).

<b>_Ongoing</b>	0 or > 0 (numeric)	Sets whether this is an "ongoing" notification. Ongoing notifications cannot be dismissed by the user, so your application must take care of canceling them. They are typically used to indicate a background task that the user is actively engaged with (e.g., playing music) or is pending in some way and therefore occupying the device (e.g., a file download, sync operation, active network connection). Default is false (0).
<b>_ShowWhen</b>	0 or > 0 (numeric)	Beginning with Android 7 (API level 26) the notification time has to be set, if you want to display it. Default is false (0).

The optional numeric variable <notification\_id\_nvar> controls and returns a notification id.

<notification_id_nvar>	Control and Result
-2	A new notification id is created by the last nine digits of the current system time in milliseconds. Returns the created Id.
-1	Repeats and returns the last used notification id.
0	Default, returns 0
Given id	Uses the given id if possible. Otherwise 0 will be used and returned.



#### Example

```
! bundle.put b, "_SmallIcon", "myIconAlphaChaneled.png"
bundle.put b, "_Largelcon", "cartman.png"
bundle.put b, "_Sound", "whee.mp3"
maxVal = 1000
progress = 333
animate = 0
bundle.put b, "_Progress", int$(maxVal) + "," + int$(progress) + "," + int$(animate)
bundle.put b, "_Color", "255,0,0" % Red
myNnotify_ID_1 = -2
Notify "title1", "subtitle", "alert", 0 , b, myNnotify_ID_1
? myNnotify_ID_1 : ?

bundle.put b, "_Sound", ""
bundle.put b, "_Color", "0,255,0" % Green
myNnotify_ID_2 = -2
Notify "title2", "subtitle", "alert", 0 , b, myNnotify_ID_2
? myNnotify_ID_2 : ?

bundle.put b, "_Largelcon", "galaxy.png"
bundle.put b, "_Color", "0,0,255" % Blue
myNnotify_ID_3 = -2
Notify "title3", "subtitle", "alert", 1 , b, myNnotify_ID_3, ret
? myNnotify_ID_3 : ?
? ret
do
until 0
```

#### **Notify.cancel {<notification\_id\_nexp>}**

Cancels all notifications triggered by your program by default.

Does not overwrite <wait\_lexp> from the command Notify, because in this case the execution of the BASIC! program is suspended until the user taps the Notify object.

If you use the optional <notification\_id\_nexp> with a number > 0 only a notification with this number will be canceled. If <notification\_id\_nexp> is 0 all notifications will be canceled like the default setting.

## PERMISSIONS

Permissions are special privileges that apps must ask for if they want to access sensitive media on your Android device.

Android devices contain so much personal information, like the exact location, contact data, and cameras that can record the user (permission protection level dangerous). Apps can not just use these unless the user tell them it is okay, because beginning with Android 6.0 "Marshmallow", API level = 23 the permission handling is more defensive. But if your program is running on a lower API-level, no restrictions will happen.

Permissions with have protection level dangerous **have be requested first** in Basic's or your application's manifest, by default these are not be granted to your app at first start after the first installation.

*Note, that normal permissions are granted at install time if requested in the manifest.*

The permissions of an app can be checked at any time. If the app is already installed, go to Settings → Apps and locate the app you want to examine. Tap an app and on the About screen, click the Permissions box or scroll down to find the list. Here you can see everything the app asks for. For a programmed link see App.Settings.

File access to Resources, Assets and the Internal file directory should not need any permissions, because these are protected areas.

It is a good practice to check the needed permissions at the first application start directly. To prevent irritations explain your requested permissions in detail, because if you trigger the DEVICE command with a bundle, you need Phone permissions. But if you do not want to make a call, you should describe that you need only device information.

Do not be surprised if the application user later changes one or more of the required permissions.

May be the user denied the permissions for camera and microphone for privacy reasons. So you should take appropriate precautions. Using permission.get is a possibility.

If you want to compile your program as an APK, Permission.checkPath is useful. You can check, is the given file path is a candidate for a file permission request.

*Note, that permissions in conjunction with Internal Directories on an External Directory or on removable SD-cards were not tested until now.*

In conjunction to deliver your program as an APK the conclusion is, ask as soon as possible for all needed permissions. Use assets and the internal directory for easy file permission control, because in this case you do not have to request for permissions.

Permissions used by AndroidManifest.xml	Level Dangerous
ACCESS_COARSE_LOCATION	✓
ACCESS_FINE_LOCATION	✓
ACCESS_LOCATION_EXTRA_COMMANDS	
ACCESS_mock_LOCATION	
ACCESS_NETWORK_STATE	
ACCESS_SUPERUSER	
ACCESS_WIFI_STATE	
BLUETOOTH_ADMIN	
BLUETOOTH	
CALL_PHONE	✓
CAMERA	✓
INSTALL_SHORTCUT	
INTERNET	
KILL_BACKGROUND_PROCESSES	
READ_CONTACTS	✓
READ_EXTERNAL_STORAGE	✓
READ_PHONE_STATE (Android 10-) or READ_PHONE_NUMBERS (Android 11+)	✓
READ_SMS	✓
READ_USER_DICTIONARY	
RECEIVE_SMS	✓
RECORD_AUDIO	✓
RUN_SCRIPT (Only to detect over APP.settings!)	✓
SEND_SMS	✓
UNINSTALL_SHORTCUT	
VIBRATE	
WAKE_LOCK	
WRITE_EXTERNAL_STORAGE	✓
WRITE_SETTINGS	

*Note, that WRITE\_EXTERNAL\_STORAGE includes the READ\_EXTERNAL\_STORAGE permission.*

### Permission.automatic <auto\_nvar>

OliBasic has an automatic permission detection and request. Which can be switched to OFF by <auto\_nvar> = 0 or to ON by <auto\_nvar> = 1 (default).

If <auto\_nvar> = 2 it is switched to ON but with no file permissions.

It is only an option to switch to OFF in case of massive data file transfer because the system delay. If possible Permission.ignore is the better option.

### Permission.checkPath <nvar>, <checkPath\_sexp>, <dump\_nexp>

If <nvar> = 1 the given file name <checkPath\_sexp> is a candidate for checking the STORAGE permission related to the shared external storage (READ\_EXTERNAL\_STORAGE, WRITE\_EXTERNAL\_STORAGE).

If <dump\_nexp> > 0 detailed information are printed.

### Permission.ignore <file\_path\_Array\$[]>

Dealing with file access permissions requires some care.

Assets and Internal directories should be ignored for permission checking.

File names including "files:///data/", "Android/data/(Application's package name)" and "asset://" will be ignored by default.

If you deal with relative file paths, it could be useful to add your own exceptions.

### Permission.get <granted\_lvar>, <permission\_sexp>

Returns <granted\_lvar> = 1 if the permission given by <permission\_sexp> is granted. Otherwise <granted\_lvar> returns 0.

### Permission.request <permission\_sexp> | Array\$[]

For Android 6.0 Marshmallow, API level = 23 and later.

Requests permissions to be granted to the running Basic-engine or your application(APK).

Level Dangerous Permission Group	Related Permissions
<b>CALENDAR</b> Used for runtime permissions related to user's calendar.	READ_CALENDAR WRITE_CALENDAR
<b>CALL_LOG</b> (Since Android 9) This permission group gives control and visibility to apps that need access to sensitive information about phone calls, such as reading phone call records and identifying phone numbers.	PROCESS_OUTGOING_CALLS READ_CALL_LOG WRITE_CALL_LOG
<b>CAMERA</b> Used for permissions that are associated with accessing camera or capturing images/video from the device.	CAMERA
<b>CONTACTS</b> Used for runtime permissions related to	READ_CONTACTS WRITE_CONTACTS

contacts and profiles on this device.	GET_ACCOUNTS
<b>LOCATION</b> Used for permissions that allow accessing the device location. OliBasic supports by default only ACCESS_FINE_LOCATION (GPS and Network). <del>If only ACCESS_COARSE_LOCATION wanted, use Permission.request-"ACCESS_COARSE_LOCATION" at app start first. Both together are not allowed.</del>	<del>(ACCESS_COARSE_LOCATION)</del> ACCESS_FINE_LOCATION
<b>MICROPHONE</b> Used for permissions that are associated with accessing microphone audio from the device.	RECORD_AUDIO
<b>PHONE</b> Used for permissions that are associated with telephony features.	ANSWER_PHONE_CALLS READ_PHONE_STATE READ_PHONE_NUMBERS CALL_PHONE ADD_VOICEMAIL USE_SIP (Until Android 8.1 PROCESS_OUTGOING_CALLS READ_CALL_LOG WRITE_CALL_LOG )
<b>SENSORS</b> Used for permissions that are associated with accessing body or environmental sensors.	BODY_SENSORS
<b>SMS</b> Used for runtime permissions related to user's SMS messages.	READ_SMS SEND_SMS RECEIVE_SMS RECEIVE_WAP_PUSH RECEIVE_MMS
<b>STORAGE</b> Used for runtime permissions related to the shared external storage.	READ_EXTERNAL_STORAGE WRITE_EXTERNAL_STORAGE
<b>Random Terminal Scripts</b>	Only to control over APP.settings

*Note, that not all table permissions are used in conjunction with Basic.*

Normal permissions are granted at install time if requested in the manifest.

If your program does not have the requested permissions the user will be presented with UI for accepting them.

Note that requesting a permission does not guarantee it will be granted and your program should be able to run without having this permission.

This command may start an activity allowing the user to choose which permissions to grant and which to reject. Hence, you should be prepared that your activity may be paused

and resumed. **Further, granting some permissions may require a restart of your application!**

When checking whether you have a permission you should use `Permission.get`.

Google Reference:

Calling this command for permissions already granted to your app would show UI to the user to decide whether the app can still hold these permissions. This can be useful if the way your app uses data guarded by the permissions changes significantly.

**But after experiences:**

The UI is only created , if the permission is not granted.

*Note, use the single string only, if will request one permission.*

*A second one or more without a pause will be ignored. Use instead an array.*

Example:

```
Permission.request "READ_PHONE_STATE"
```

See also:

`Permission.get`, `App.settings`, `WiFi.info`

**WiFi.info {{<SSID\_svar>},{, <BSSID\_svar>},{, <MAC\_svar>},{, <IP\_var>},{, <speed\_nvar>}}**

Gets information about the current Wi-Fi connection and places it in the return variables. All of the parameters are optional; use commas to indicate omitted parameters (see Optional Parameters). The table shows the available data:

Variable	Type	Returned Data	Format
SSID	String	SSID of current 802.11 network	"name" or hex digits (see below)
BSSID	String	BSSID of current access point	xx:xx:xx:xx:xx:xx (MAC address)
MAC	String	MAC address of your WiFi	xx:xx:xx:xx:xx:xx
IP	Numeric or String	IP address of your WiFi	Number or octets (see below)
speed	Numeric	Current link speed in Mbps	Number

Format notes:

- SSID: If the network is named, the name is returned, surrounded by double quotes. Otherwise the returned name is a string of hex digits.
- IP: If you provide a numeric variable, your Wi-Fi IP address is returned as a single number. If you provide a string variable, the number is converted to a standard four-octet string. For example, the string format 10.70.13.143 is the same IP address as the number -1887287798 (hex 8f82460a).

If you need <SSID\_svar> and <BSSID\_svar> your device needs on newer Android systems access to Fine Location Permissions. On devices below Android 6 this code will also run, because Permission commands will be ignored.

```
PERMISSION.GET granted, "ACCESS_FINE_LOCATION"
```

```
IF !granted THEN PERMISSION.REQUEST "ACCESS_FINE_LOCATION"
```

```
WIFI.INFO ssid$, bssid$, mac$, ip$, speed
```

```
PRINT ssid$, bssid$, mac$, ip$, speed
```

### CHR\$(<nexp>, ...)

Return the character string represented by the values of list of numerical expressions. Each <nexp> is converted to a character. The expressions may have values greater than 255 and thus can be used to generate Unicode characters.

PRINT CHR\$(16\*4 + 3)                      % Hexadecimal 43 is the character "C":

Prints: **C**

PRINT CHR\$(945, 946)                      % Decimal for the characters alpha and beta:

Prints: **αβ**

PRINT CHR\$(128194)                      % 32-bit character open folder:

Prints: **📁**

PRINT CHR\$(55357,56514)                  % Creates also a 32-bit char. by two 16-bit char.:

Prints: **📁**

Android uses 16-bit characters so you should note, that a 32-bit character counts two 16-bit characters.

LEN("📁") % Returns 2

See also

[https://www.w3schools.com/charsets/ref\\_html\\_utf8.asp](https://www.w3schools.com/charsets/ref_html_utf8.asp) (To choose!)

<https://www.fileformat.info/info/unicode/char/search.htm> (To get the 2 dec. values!)

### UCODE(<sexp>{, <index\_nexp>})

Returns the Unicode value of one character of <sexp>. By default, it is the value of the first character. You can use the optional <index\_nexp> to select any character. The index of the first character is 1.

If <sexp> is an empty string ("") the value returned will be ~~65536 (one more than the largest 16-bit Unicode value)~~ 1000000.0. If the selected character of <sexp> is a valid ASCII character, this function returns the same value as **ASCII()**.

In case of 32-bit characters UCODE("📁", 1) returns 55357 and UCODE("📁", 2) returns 56514.

### UCODE32(<sexp>)

Returns the Unicode value of the first 16- or 32-bit character of <sexp>.

If <sexp> is an empty string ("") the value returned will be 1000000.0. If the selected character of <sexp> is a valid ASCII character, this function returns the same value as **ASCII()**.

In case of 32-bit characters UCODE32("📁") returns 128194.



**QR.create.svg <fileName\_sexp>, <text\_sexp>{{, <level\_sexp>},  
<bundlePtr\_nvar>}**

Creates a SVG vector file with a QR code.

The vector file name will be defined by <fileName\_sexp>. The QR code will be created from the text given by <text\_sexp>.

The level of error correction will be set optionally by <level\_sexp>. Default is "\_Medium".

Level L ("\_Low") 7% of data bytes can be restored.

Level M ("\_Medium") 15% of data bytes can be restored.

Level Q ("\_Quartile") 25% of data bytes can be restored.

Level H ("\_High") 30% of data bytes can be restored.

Optional Bundle content from <bundlePtr\_nvar>

Table of SVG control options		
Key	Value	Description
<b>_Border</b>	numeric	Sets the border around the QR code. Default is 1.
<b>_PatternColor</b>	{Alpha,}Red,Green,Blue (comma delimited string) or _{Alpha,}ColorName ({comma delim.} string) or #{hn}hn hnhn (hex. string)	The Alpha channel will be not interpreted in opposite to _BackgroundColor.
<b>_BackgroundColor</b>	{Alpha,}Red,Green,Blue (comma delimited string) or _{Alpha,}ColorName ({comma delim.} string) or #{hn}hn hnhn (hex. string)	
<b>_Size</b>	String	Sets the size of the quadratic QR code. The measurement units can be nothing, pxl, in, cm and mm. If the string is empty, the width and height parameters of the SVG header will be deleted. In this case Webview and HTML browsers scale the SVG file within the display borders. But this file cannot be load by OliBasic. Because it needs width and height. Example: "50mm"

Example:

```
! Confugration of the QR code result
Bundle.put b, "_Border", 3
Bundle.put b, "_PatternColor", "_DarkRed"
Bundle.put b, "_BackgroundColor", "_LightPink"
Bundle.put b, "_Size", "19cm"

Gr.open "_Gray", 1, 1
Gr.screen mGrWidth, mGrHeight
Qr.create.svg "test.svg", "Hello World!", "_High", b
Gr.bitmap.load bmpPtr, "test.svg", mGrWidth, mGrHeight

Gr.statusbar inset, w
Gr.bitmap.draw oPtr1, bmpPtr, 0, inset
Gr.render

Do
  Pause 100
Until 0
End
```

**IDEAS**

### Program.info <nexp>|<nvar>

Returns a Bundle that reports information about the currently running program. If you provide a variable that is not a valid Bundle pointer, the command creates a new Bundle and returns the Bundle pointer in your variable. Otherwise it writes into the Bundle your variable or expression points to.

The bundle keys and possible values are in the table below:

Key	Type	Value
<b>_BasPath</b>	String	Full path + name of the program currently being executed. The path is relative to BASIC!'s "source/" directory. <a href="#">See also File.root</a>
<b>_BasName</b>	String	Name of the program currently being executed.
<b>_SysPath</b>	String	Full path to the BASIC!'s private file storage directory. The path is relative to BASIC!'s "data/" directory. <a href="#">See also File.root</a>
<b>_UserApk</b>	Numeric (Logical)	Returns 1.0 (true) if the current program is being run from a standalone user-built APK (Appendix D). Returns 0.0 (false) if the program is being from the BASIC! Editor or a Launcher Shortcut (Appendix C).
<b>_LauncherStart</b>	Numeric (Logical)	Returns 1.0 (true) if the current program is being run from a Launcher Shortcut (Appendix C) or an intent. Returns 0.0 (false) in other cases.
<b>_PackageName</b>	String	<a href="#">package_id</a>
<b>_AppVersion</b>	String	Application Version (from the AndroidManifest.xml)
<b>_AppVersionCode</b>	Numeric	Application Version Code (from the AndroidManifest.xml)
<b>_Build</b>	String	Returns the Build of the underlying OliBasic Version
<b>_MyProcessId</b>	String	<a href="#">my_process_id</a>
<b>_TotalUserRam</b>	Numeric	The total memory accessible by the kernel. This is basically the RAM size of the device, not including below-kernel fixed allocations like DMA buffers, RAM for the baseband CPU, etc.
<b>_AvailableRam</b>	Numeric	The available memory on the system. This number should not be considered absolute: due to the nature of the kernel, a significant portion of this memory is actually in use and needed for the overall system to run well.
<b>_ThresholdRam</b>	Numeric	The threshold of available memory at which we consider memory to be low and start killing background services and other non-extraneous processes.
<b>_HeapLimit</b>	Numeric	Return the approximate per-application memory class of the current device. This gives you an idea of how hard a memory limit you should impose on your application to let the overall system work best.
<b>_NativeHeap</b>	Numeric	Returns the (dynamic) size of the native heap.

Key	Type	Value
<b>_NativeHeapAllocated</b>	Numeric	Returns the amount of allocated memory in the native heap.
<b>_NativeHeapFree</b>	Numeric	Returns the amount of free memory in the native heap.

For example, assume:

- You are using the default <pref base drive>
- You downloaded a file called "my\_program.bas" to the standard Android Download directory.
- You used the BASIC! Editor to load and run the downloaded program.

Then the returned values would be as follows:

Key	Value
<b>_BasPath</b>	.././Download/my_program.bas
<b>_BasName</b>	my_program.bas
<b>_SysPath</b>	.././././././data/data/com.rfo.basic
<b>_UserApk</b>	0.0
<b>_PackageName</b>	com.rfo.basic
<b>_MyProcessId</b>	21615
<b>_AppVersion</b>	3.00
<b>_AppVersionCode</b>	3030.0
<b>_Build</b>	3.00 Preview30n
<b>_TotalUserRam</b>	2759.0 in megabytes.
<b>_AvailableRam</b>	1370.0 in megabytes.
<b>_ThresholdRam</b>	216.0 in megabytes.
<b>_HeapLimit</b>	192.0 in megabytes.
<b>_NativeHeap</b>	15.89... in megabytes.
<b>_NativeHeapAllocated</b>	12.19... in megabytes.
<b>_NativeHeapFree</b>	3.69... in megabytes.

**SysPath:** BASIC! normally keeps programs and data in its *base directory* (see **Working with Files**, later in this manual). The base directory is in public storage space (**external**). BASIC! programs also have access to a private (**internal**) storage area. Your program can create a sub directory within the SysPath directory and store private files there. Note that if you uninstall BASIC!, any files in private storage will be deleted.

SysPath is of particular interest to you if you build a BASIC! program as an application in a standalone apk, as described in Appendix D. See **Modifications to the AndroidManifest.xml File Permissions**.

You can extent the heap size by inserting `android:largeHeap="true";` in the APK's AndroidManifest.xml File.

## Zip Commands now in RFO-Basic 1.91 included

### ZIP File I/O

The ZIP file I/O commands work with compressed files. ZIP is an archive file format that stores multiple directories and files, using a method of lossless data compression to save file space.

Use **Zip.dir** to get an array containing the names of all of the directories and files in an archive. Use the file names with **Zip.read** to extract files from the archive. **Zip.read** can not extract a directory. Use **Zip.write** to put files in a new archive. You can overwrite an existing ZIP file, but you cannot replace or add entries.

### Zip.dir <path\_sexp>, Array\$[] {,<dirmark\_sexp>} {,<timeStamp\_nexp>}

Returns the names of the files including the internal path beginning with ZIP root and directories inside the ZIP file which is located at <path\_sexp>. The path is relative to "<pref base drive>/rfo-basic/data/".

The names are placed into Array\$[]. The array is sorted alphabetically with the directories at the top of the list. If the array exists, it is overwritten, otherwise a new array is created. The result is always a one-dimensional array.

A directory is identified by a marker appended to its name. The default marker is the string "(d)". You can change the marker with the optional directory mark parameter <dirmark\_sexp>. If you do not want directories to be marked, set <dirmark\_sexp> to an empty string, "".

If the directory is empty, Zip.dir returns an array with one item and a string with one space (" ") in it.

Options of <timeStamp\_nexp>:

- 0 no time stamp (default), sorted alphabetically
- 1 with time stamp as time in milliseconds + ":" + file name, but unsorted
- 2 with time stamp as time in milliseconds + ":" + file name, sorted in ascending order
- 3 with time stamp as time in milliseconds + ":" + file name, sorted in descending order

### Zip.open {r|w|a}, <file\_table\_nvar>, <path\_sexp>

**Deprecated, use ZIP.Files and ZIP.Extract if possible.**

The ZIP file specified by the path string expression <path\_sexp> is opened. The path is relative to "<pref base drive>/rfo-basic/data/".

The first parameter is a single character that sets the I/O mode for this file:

Parameter	Mode	Notes
r	read	File exists: Reads from the start of the file. File does not exist: Error (see below).
w = a	write	File exists: Writes from the start of the file. Writes over any existing data. File does not exist: Creates a new file. Writes from the start of the file.

Note: Unlike **Text.open** and **Byte.open**, **Zip.open** does not support an append mode.

A file table number is placed into the numeric variable <file\_table\_nvar>. This value is for use in subsequent **Zip.read**, **Zip.write**, or **Zip.close** commands.

If there was an error opening the ZIP file, <file\_table\_nvar> is set to -1 with details available from the **GETERROR\$()** function.

Zip.open has no access to **assets** and **resources** in conjunction with APKs. If you want to read a ZIP that is in assets, you must first copy it from assets to an internal or external file system. Then you can open this new file with Zip.extract.

See also Byte.copy

**Zip.files** FilesArray\$, EntriesArray\$, <zipFile\_path\_sexp>{{ <no\_compr\_sexp>}, <compr\_type\_sexp>}

Directory and file paths specified by FilesArray\$[] will be stored into a Zip file defined by <zipFile\_path\_sexp>. The paths placed into EntriesArray\$[] start at the root of the Zip file. Directory and file names can be changed from the original in the entries.

The date and time of the source files will be stored also.

Keep in mind, that the directories have to be specified by the Zip file separately.

File Zipping needs time. Take care if you want to zip your complete Data directory.

Sometimes it makes no sense to compress a compressed file. If the content of <no\_comp\_sexp> contains strings like "jpg/gif/mp3 ..." files described in the EntriesArray\$[] with an ending like ". jpg", ". gif", ".mp3" ... will be not compressed. The delimiter is the slash "/", because the comma can be part of a file name.

If you need fast access and no compression but an archive a "" within <no\_comp\_sexp> will skip the compression of all files.

The system constants in the optional <comp\_rate\_sexp> enables easy access to compression types:

\_SYNC\_FLUSH, \_NO\_FLUSH, \_FULL\_FLUSH, \_HUFFMAN\_ONLY, \_FILTERED,  
\_DEFLATED, \_DEFAULT\_STRATEGY, \_DEFAULT\_COMPRESSION,  
\_BEST\_COMPRESSION, \_BEST\_SPEED, \_NO\_COMPRESSION

Note that \_SYNC\_FLUSH, \_NO\_FLUSH, \_FULL\_FLUSH are available beginning with Android 4.4 (KitKat). Earlier versions use in these cases \_DEFAULT\_COMPRESSION by default.

Default is "" that equals \_DEFAULT\_COMPRESSION.

**Zip.extract** <destination\_path\_sexp>, <zipFile\_path\_sexp>{{, <skip\_overwrite\_nexp>}, ToExtractArray\$[]}

Extracts the content of the Zip file specified by <zipFile\_path\_sexp> into the destination defined by <destination\_path\_sexp>. The destination directory will be created automatically. Older files and directories will be overwritten. The last modified date of the source files will be inserted also.

In the event of an error please note that the directories in the Zip file must be specified separately.

The optional <skip\_overwrite\_nexp> controls the overwriting of existing files. If <skip\_overwrite\_nexp> is 0 overwriting will be done in any case, if it is 1 overwriting will be skipped if the last modified date of the new file is older and if it is 2 overwriting will be skipped in any case. Default is 0.

The file paths described by ToExtractArray\$[] will be extracted if the Zip file contains the right counterpart. (Zip.dir <path\_sexp>, Array\$[] , "")

Be sure, that the needed directories are created before. ????

Zip.extract has no access to **assets** and **resources** in conjunction with APKs. If you want to read a ZIP that is in assets, you must first copy it from assets to an internal or external file system. Then you can open this new file with Zip.extract. ??? **Could work also.**  
See also Byte.copy

See also File.move

Example:

```
FILE.MKDIR "Suitcase"
BYTE.OPEN r, ftb, "cartman.png"
BYTE.COPY ftb, "Suitcase/cartman.png" % LastModified is the current date and time now.
BYTE.CLOSE ftb
FILE.MKDIR "Suitcase/insects"
BYTE.OPEN r, ftb, "fly.gif"
BYTE.COPY ftb, "Suitcase/insects/fly.gif"
BYTE.CLOSE ftb
FILE.MKDIR "Suitcase/things"
path$ = "Suitcase"
FILE.DIR path$, mDirArray$, "", 0 , 1 , "_DF" % In case of Zip.files only "_D" or "_DF"
ARRAY.LENGTH al, mDirArray$[]
DIM mEntries$(al)
DIM mDirArrayZip$(al)
FOR i = 1 TO al
    mDirArrayZip$(i)= path$ + "/" + mDirArray$(i)
    mEntries$(i) = mDirArrayZip$(i) % Extracted as Extracted/Suitcase/insects/fly.gif
    ! mEntries$(i) = mDirArray$(i) % Extracted as Extracted/insects/fly.gif
NEXT
ZIP.FILES mDirArrayZip$, mEntries$, "my.zip"
ZIP.EXTRACT "Extracted", "my.zip"
ZIP.DIR "my.zip", zipArray$[]
DEBUG.ON : DEBUG.DUMP.ARRAY zipArray$[]
```



## Good to know

### Something about AndroidManifest.xml

If you want only encoded IP connections like Https instead of Http change `android:usesCleartextTraffic` to false.

### Something about touch events

Android triggers touch events at a distance of at least 16 to 18 milliseconds.

<b>Standard Values</b> Some of them can be changed by the user in global preferences. All device independent pixels (DIPs) are in conjunction to the Android standard 160 DPI resolution.		
Tap Timeout	100	Duration in milliseconds we will wait to see if a touch event is a tap or a scroll. If the user does not move within this interval, it is considered to be a tap.
Double Tap Timeout	300	Duration in milliseconds between the first tap's up event and the second tap's down event for an interaction to be considered a double-tap.
Long Press Timeout	400	Duration in milliseconds before a press turns into a long press
Maximum Fling Velocity	8000	Maximum velocity to initiate a fling, as measured in DIPs per second.
Minimum Fling Velocity	50	Minimum velocity to initiate a fling, as measured in DIPs per second.
Touch Slop	8	Distance in DIPs a touch can wander before we think the user is scrolling

### NaN (Not a Number) or Infinity values (IEEE 754)

These are floating point values in our case from type Double.

If you try SQR (-1) or 0/0 you get a NaN, because it is an undefined operation.

But if you try 1/0 you get an Infinity, because it is an infinite result.

Or you get these values per definition like VAL("Infinity"), VAL("-Infinity") and VAL("NaN").

This differs from older Basic implementations and in some parts also from RFO-Basic 1.91.

That should not have a negative impact on older code. Hopefully in this development stage, you will only get a runtime error if you want to handle these values and functions, where input values must be of the type Integer or Long.

Functions which use BigDecimal inside do not deal with NaN (Not a Number) or Infinity values.

Examples: BigD(decimal) command group, List.join

Functions which use <lexp> and <nexp> interpret these values as 0.

For detection use Is\_NaN and Is\_Infinite. The last one returns -1 if the value is negative.

#### !NaN Example

```
DIM x[1]
x[1] = 1
ARRAY.STD_DEV nanV, x[ ] % Returns NaN in all Basic! versions.
IF nanV THEN ? "true" : ELSE ? "false" % → true But that is wrong, see below
IF Is_Number("NaN") THEN ? "true" : ELSE ? "false" % → true
IF Is_NaN(nanV) THEN ? "true" : ELSE ? "false" % → true
IF nanV = nanV THEN ? "true" : ELSE ? "false" % → false
IF nanV <> nanV THEN ? "true" : ELSE ? "false" % → true
IF nanV < 0 THEN ? "true" : ELSE ? "false" % → false
IF nanV > 0 THEN ? "true" : ELSE ? "false" % → false
```

#### !Infinity Example

```
!infiV = -1/0 %Returns -Infinity.
infiV = val("-Infinity") % Returns -Infinity in all Basic! versions.
? infiV
IF infiV THEN ? "true" : ELSE ? "false" % → true
IF Is_Number("-Infinity") THEN ? "true" : ELSE ? "false" % → true
IF Is_Infinite(infiV) THEN ? "true" : ELSE ? "false" % → true
? Is_Infinite(infiV) % → -1
IF infiV = infiV THEN ? "true" : ELSE ? "false" % → true
IF infiV <> infiV THEN ? "true" : ELSE ? "false" % → false
IF infiV < 0 THEN ? "true" : ELSE ? "false" % → true
IF infiV > 0 THEN ? "true" : ELSE ? "false" % → false
```

#### Use instead of

```
IF nanV THEN ? "true" : ELSE ? "false" % → true But that is wrong
IF !Is_NaN(nanV) & nanV THEN ? "true" : ELSE ? "false" % → false That is right
WHILE nanV % → true But that is wrong
WHILE !Is_NaN(nanV) & nanV % → false That is right
UNTIL nanV % → true But that is wrong
UNTIL !Is_NaN(nanV) & nanV % → false That is right
```

## Sum of Arrays

```
nanV = VAL("NaN")
? nanV
IF IS_NAN(nanV) THEN ? "true" : ELSE ? "false" % → true
infiV = VAL("-Infinity") % Returns -Infinity in all Basic! versions.
? infiV
IF infiV THEN ? "true" : ELSE ? "false" % → true
```

```
ARRAY.LOAD exampl[], 1, 2, 3
ARRAY.SUM s, exampl[]
PRINT s % → 6
```

```
ARRAY.LOAD exampl[], 1, 2, 3, nanV
ARRAY.SUM s, exampl[]
PRINT s % → NaN
```

```
ARRAY.LOAD exampl[], 1, 2, 3, infiV
ARRAY.SUM s, exampl[]
PRINT s % → -Infinity
```

```
ARRAY.LOAD exampl[], 1, 2, 3, nanV, infiV
ARRAY.SUM s, exampl[]
PRINT s % → NaN
```

## Comparison with NaN

A comparison with a NaN always returns an *unordered result* even when comparing with itself. The comparison predicates are either signaling or non-signaling on quiet NaN operands; the signaling versions signal the invalid operation exception for such comparisons. The equality and inequality predicates are non-signaling so  $x = x$  returning false can be used to test if  $x$  is a quiet NaN. The other standard comparison predicates are all signaling if they receive a NaN operand, the standard also provides non-signaling versions of these other predicates. The predicate *isNaN( $x$ )* determines if a value is a NaN and never signals an exception, even if  $x$  is a signaling NaN.

Comparison between NaN and any floating-point value  $x$  (including NaN and  $\pm\infty$ )

Comparison	$\text{NaN} \geq x$	$\text{NaN} \leq x$	$\text{NaN} > x$	$\text{NaN} < x$	$\text{NaN} = x$	$\text{NaN} \neq x$
Result	Always <b>False</b>	Always <b>False</b>	Always <b>False</b>	Always <b>False</b>	Always <b>False</b>	Always <b>True</b>

See also

[https://docs.oracle.com/cd/E19957-01/806-3568/ncg\\_intro.html#110](https://docs.oracle.com/cd/E19957-01/806-3568/ncg_intro.html#110)

[https://docs.oracle.com/cd/E19957-01/806-3568/ncg\\_goldberg.html](https://docs.oracle.com/cd/E19957-01/806-3568/ncg_goldberg.html)



## Fixes

BT.READ.BYTES and BT.UTF\_8 memory overflow on longer messages

Fixed

GRABFILE does not close its input stream.

Fixed

FN.RTN within loops does not remove the outdated stack entries.

Fixed

CLIPBOARD.GET gets an empty result on Android 10+ if operating system's security check is not finished.

Fixed by a pause of 100 milliseconds before execution

The Save dialog allows multiple lines for a file name.

Fixed

FTP.RENAME returns in every case an error.

Fixed

GR.MODIFY Type "group" and parameter "list" return a list error.

Fixed

Crash on some devices at Gr.Rotate.End before Gr.Rotate.Start

Fixed by ignoring Gr.Rotate.End in this case

The HTTP commands do not send all parameters.

Fixed

GR.TOUCH2 and GR.BOUNDED.TOUCH2 <touched> returns 0.0 instead of 1.0 if finger 1 is lifted up. When the second finger already touches the display, a second touch of the first finger initiates a rebound of the second finger without raising it!

Fixed

Calling Basic! by a modern file browser fails if it uses a document path beginning with "content://".

Fixed

WIFI.INFO Android issue returns wrong results for the MAC address.

Fixed

GR.BITMAP.CROP Crashes, if wished bounds are outside the given. [GitHub#268](#)

Fixed

"&=" was dropping into "|=" op immediately after, giving wrong result. [Humpty#0243](#)

Fixed

GR.ARC, GR.MODIFY Some devices do not ignore NaN or Infinity values. Now you get a runtime error, if Gr.Arc (angels) gets NaN or Infinity values. [GitHub#267](#)

Fixed

GR.TOUCH <touched> returns sometimes 1.0 instead of 0.0 in special cases.

Fixed

GR.ARRAY.TOUCH <count\_nvar> returns 1.0 instead of 0.0.

Fixed

GR.LIST.TOUCH <count\_nvar> returns only 0.0.

Fixed

GR.CAMERA.\_\_\_\_SHOOT commands return no valid bitmap pointer.

Fixed

CONSOLE.SAVE in graphics or HTML mode returns an empty file.

Fixed by a runtime error message

EMAIL.SEND the last four optional arguments are working as a group instead of single arguments

Fixed

SENSORS.READ crashes on Android 8 if a sensor returns only one or two arguments [GitHub#265](#)

Fixed

Ctrl A, C, V, X

Fixed

Byte.write.buffer can only write one time before Byte.close

Fixed

Split returns the same as Split.all

Fixed

GrabURL reads cache [GitHub#264](#)

Fixed

FILE.EXISTS fails, if a variable and a String are part of the second argument

Fixed

ROUND fails often, because imprecise double import [GitHub#262](#)

Fixed

BIGD.FROMDOUBLE imprecise double import

Fixed

BIN\$ Input of a NaN (Not a Number) or Infinity Value returns 0 [GitHub#260](#)

Fixed with a return of "NaN"

INT Input of a NaN (Not a Number) or Infinity Value returns 0.0 [GitHub#259](#)

Fixed with a return of "NaN"

INT\$ Input of a NaN (Not a Number) or Infinity Value returns 0 [GitHub#259](#)

Fixed with a return of "NaN"

HEX\$ Input of a NaN (Not a Number) or Infinity Value returns 0 [GitHub#259](#)

Fixed with a return of "NaN"

**OFT\$ Input of a NaN (Not a Number) or Infinity Value returns 0** [GitHub#259](#)

**Fixed with a return of "NaN"**

**BIN\$ Input of a NaN (Not a Number) or Infinity Value returns 0** [GitHub#259](#)

**Fixed with a return of "NaN"**

**USING\$ Input of a NaN (Not a Number) or Infinity Value for arguments needing Integer or Long values returns 0** [GitHub#258](#)

**Fixed, Double values return NaN, Integer values throw runtime errors**

**ROUND Input of a NaN (Not a Number) or Infinity Value returns 0.0 or crashes if the scale is specified.** [GitHub#257](#)

**Does not round if only the scale is specified.**

**Fixed**

**The Editor stops until REPLACE ALL in Search is used with empty search string**

[GitHub#256](#)

**Fixed**

**BYTE.WRITE.BUFFER is very slow** [GitHub#255](#)

**Improved**

**GR.CAMERA. \_\_\_SHOOT The file name extension from "image.png" is wrong in a case of jpeg compression** [GitHub#254](#)

**Fixed**

**STT.RESULTS list variable does not reset at program start** [GitHub#253](#)

**Fixed**

**VOLKEYS.ON / .OFF initialization and logic backward referenced** [GitHub#251](#)

**Fixed**

**SENSORS.LIST names not all 20 standard sensor types.** [GitHub#250](#)

**Fixed**

**BIGD.ROUND returns 0**

**Fixed**

**BUNDLE.PUT needs expressions in brackets like (n+1) or (h\$+p\$)**

**Fixed**

**BUNDLE.GET detects no number array**

**Fixed**

**ZIP.DIR ..., <dirmark\_sexp> is not used** [GitHub#246](#)

**Fixed by deleting parameter ~~rewriting~~**

**BUNDLE.PUT sends wrong error message if array does not exists**

**Fixed**

**Carriage Return characters in program code are misinterpreted** [GitHub#243](#)

**Fixed by ignoring**

**APP.START Package Name and Component Name** [GitHub#240](#)

**Fixed**, but there is still an internal design issue through sending unneeded data, which triggers sometimes unexpected results. Today with APP.SAR you are on the safe side.

**PRINT command before CLS : PRINT "Text" returns empty Screen** [GitHub#239](#)

**Fixed**

**Editor SAVE File name fails if ".bAs" file extension contains uppercase characters #**

**Fixed to ".bas" in any case**

**Some strange behaviors interacting with other BASIC! instances mainly on lame devices** [GitHub#29, #209, #238](#)

**Fixed**

**SOCKET.CLIENT.READ.FILE result** [GitHub#237](#)

**Fixed**

**SOCKET.CLIENT.WRITE.BYTES Fails after first call. At the second call the socket is disconnected.** [GitHub#237](#)

**Fixed**

**SOCKET.CLIENT.WRITE.FILE result** [GitHub#237](#)

**Fixed**

**GrabURL long time stability**

**Improved**

**FRAC() does not work correctly with big numbers like 4.5678879E8** [GitHub#234](#)

**Fixed** *Be aware, with Double you get maximal only 15 correct digits.*

**DEBUG.DUMP.BUNDLE returns an error**

**Fixed**

**RUN activate some enhancements also for APK mode**

**Fixed**

**FILE.ROOT sometimes returns null instead of ""**

**Fixed**

**BUNDLE.SAVE and BUNDLE.LOAD sometimes a key is missed**

**Fixed**

**RUN "" crashes BASIC!** [GitHub #216](#)

**See #218 below**

**Sometimes SELECT crashes, if too many data in List or Array**

**The intent transfer memory issue solved**

**Global Value Backdoor, if interrupt happens**

**Fixed with Command Groups LOCALS and GLOBALS,**

**Improved with the support of nested functions**





## Other enhancements

### Add full screen support for HTML5 videos [GitHub #224](#)

Suggestion borrowed from Nicolas Mougino

### Prevent BASIC! halt in case of unhandled Intent [GitHub #221](#)

Suggestion borrowed from Nicolas Mougino

### Prevent user APK crashing b/c of bad/missing permission [GitHub #220](#)

Suggestion borrowed from Nicolas Mougino

### Add support for an APK to register file extension(s) [GitHub #219](#)

Nicolas Mougino suggested a command `COMMAND$()`,

but this is also `retData` in

`Bundle.in <recAction_sexp>, <retData_svar>, <retBundleIndex_nvar>`

Android's `dataExtra` accepts only URLs with the query separator ("?" ), fragment separator ("#" ).

Simple command line options like `-s`, `-help` etc. are not allowed.

Use `DECODE$` with the type "URL" and the Qualifier "charset".

Workaround:

```
FN.DEF Command$()
  Bundle.in recAction$, retData$, retBundleIndex
  FN.RTN DECODE$ ("URL","charset", retData$)
FN.END
ftn$ = Command$()
```

### RUN command to restart current program [GitHub #218](#)

An idea from Nicolas Mougino, but use a little different way.

### Command `PROGRAM.INFO` [GitHub #217](#)

A Suggestion from Nicolas Mougino, ~~but use a some different values.~~

See also: `File.root` and `App.installed`

### Launching in Editor Mode

At launching with a given program path and an Intent Extra with a key named `"_BASIC!"` and a String expression `"_Editor"` the BASIC! program starts in the Editor mode.

### Launching the Editor at an exact position

At launching the Editor with a given program path and an Intent Extra with a key named `"_BASIC!"` and a String expression `"_Editor?start=<nexp?>end=<nexp?>"` the BASIC! opens the Editor optionally marking the given area.

## Broadcast on Runtime Error

On Runtime Error a Broadcast is send.

To receive this Broadcast with BASIC! use

OnBroadcast:

```
action$ = BasicEnginePackageName$ + ".broadcast.ERROR"
```

```
BROADCAST.IN retAction$, retData$, retBundleIndexF
```

```
IF retAction$ = action$
```

```
  BUNDLE.GB retBundleIndexF, "_Error", retBundleIndex
```

```
  BUNDLE.GET retBundleIndex, "_RuntimeError", runtimeError$
```

```
  PRINT "_RuntimeError: "; runtimeError$
```

```
  BUNDLE.GET retBundleIndex, "_ErrorText", errorText$
```

```
  PRINT "_ErrorText: "; errorText$
```

```
  BUNDLE.GET retBundleIndex, "_ErrorLog", errorLog$
```

```
  PRINT "_ErrorLog: "; errorLog$
```

```
BUNDLE.GET retBundleIndex, "_LineNumber", lineNumber$
```

```
PRINT "_LineNumber: "; lineNumber$
```

```
  BUNDLE.GET retBundleIndex, "_ExecutionIndex", lineNumber$
```

```
  PRINT "_ExecutionIndex: "; executionIndex$
```

```
  BUNDLE.GET retBundleIndex, "_LastCharacter", lineNumber$
```

```
  PRINT "_LastCharacter: "; lastChar$
```

```
  BUNDLE.GET retBundleIndex, "_LostContext", lostContext$
```

```
  PRINT "_LostContext: "; lostContext$
```

```
  BUNDLE.GET retBundleIndex, "_PackageName", packageName$
```

```
  PRINT "_PackageName: "; packageName$
```

```
ENDIF
```

Broadcast.resume

## Overwriting bitmaps with valid pointers

Now, bitmaps with a valid pointer will be overwritten.

Until OliBasicXXI, some GR commands always generate new entries in a Java list containing all the bitmaps. The list index therefore increases to very high values.

1. Thus the speed slows down.

2. The bitmap had to be deleted before the pointer was used again.

## Console.Save in graphics or HTML mode

Now, Console.Save returns an error if the program is still in graphics or HTML mode.

Use Gr.close and HTML.close before!

## Changing the Base Directory to the internal protected path

If OliBasic will be new installed the Base Directory is in the internal protected path.

This is the result of the new Scoped Storage protection.

## Changing the Base Directory by the Preferences menu

In case of Android 10 or lower you can change to the old location.

### A FTP server is part of the IDE

An independent FTP server can be started within the main menu. The FTP port have to be set by the Preferences Menu entry "FTP server port" before starting the first time. The port 2121 is recommended. Normally FTP ports begins with 21.

### More comment signs

Like Java, JavaScript and other programming languages these comment signs are also supported.

// - Single Line Comment and Middle of Line Comment

/\* - Block Comment,

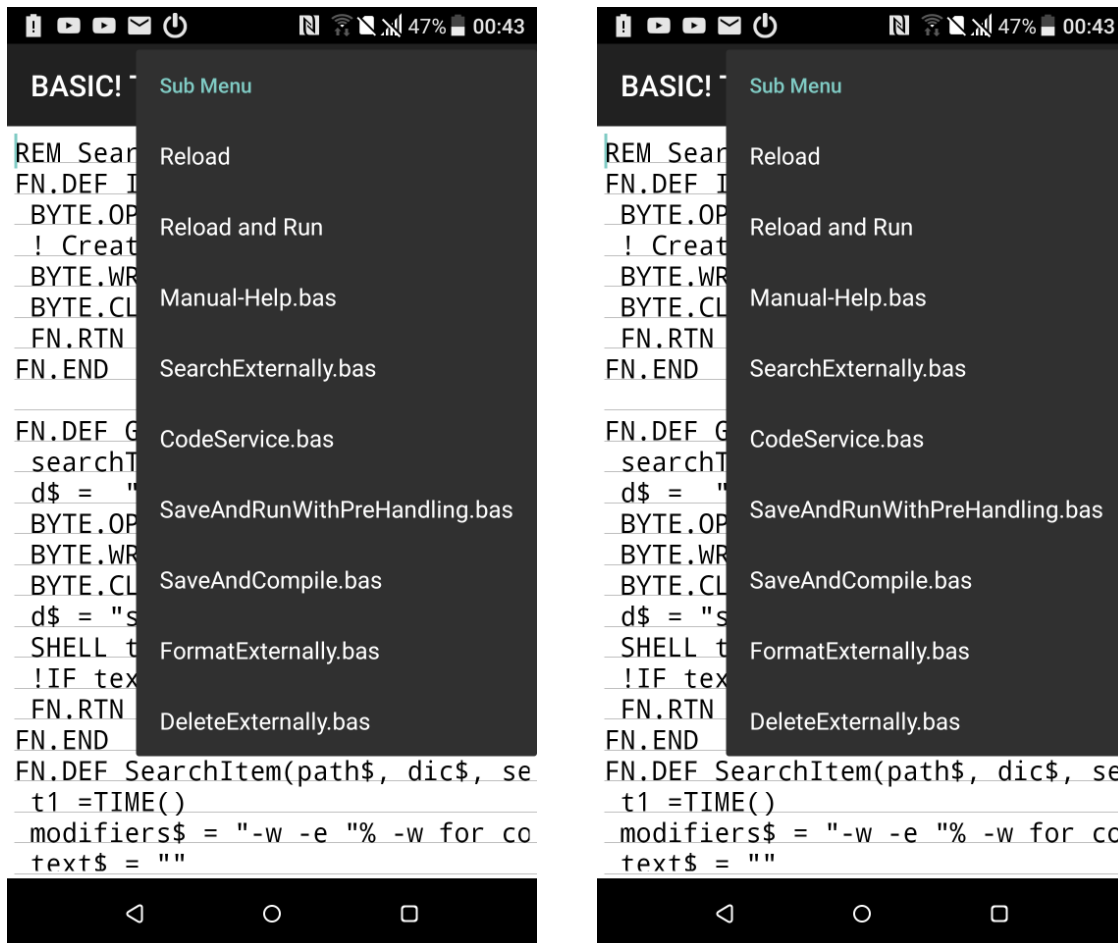
\*/ - Block Comment

### Automatic word completions or -corrections [GitHub #247](#)

Automatic word completions or -corrections are **switched to off**, because you get in trouble with the value name "thi" that returns maybe "think" after inserting a space character.

### Editor enhancements [GitHub #](#)

The Editor supports **a sub menu and keycodes** for special commands.



The entry SUB MENU ICON on action bar in Preferences/Menu\_Items\_on\_Action\_bar, supports direct access to the SubMenu.

Menu Action, bas files in source/ <b>Service_Programs/</b>	Menu	Fn Keys	Key Combination
Manual-Help.bas	Sub Menu	F1	Ctrl + Q (Query Help)
Commands		F2	Alt + Ctrl + Q
Search		F3	Ctrl + F (Find)
SearchExternally.bas	Sub Menu	Ctrl + F3	Alt + Ctrl + F
Load		F4	Ctrl + O (Open)
Reload	Sub Menu	Ctrl + F4	Ctrl + R (Reload)
Save and Run		F5	Ctrl + G (Go)
SaveAndRunWithPreHandling.bas	Sub Menu	Ctrl + F5	Alt + Ctrl + G
Save		F6	Ctrl + S (Save)
SaveAndCompile.bas	Sub Menu	Ctrl + F6	Alt + Ctrl + S
Run		F7	Ctrl + 7 (/)
CodeService.bas	Sub Menu	Ctrl + F7	Alt + Ctrl + 7
Format		F8	Ctrl + 3 (§)
FormatExternally.bas	Sub Menu	Ctrl + F8	Alt + Ctrl + 3
Clear		F9	Ctrl + N (New)
Delete		F10	Ctrl + D (Delete)
DirectoriesFiles.bas	Sub Menu	Ctrl + F10	Alt + Ctrl + D
Previous	Sub Menu	F11	Ctrl + P (Previous)
Preferences		Ctrl + F11	Alt + Ctrl + P (Preferences)
Load and Run		F12	Alt + Ctrl + O
Reload and Run	Sub Menu	Ctrl + F12	Alt + Ctrl + R

If a Sub Menu item starts with "Reload" the last loaded/saved Basic! program code will be reloaded. It is independent under which circumstances Basic! was be closed.

If a Sub Menu item ends with ".bas" it is linked to an equal named Basic! program in the source sub folder Service\_Programs.

If a Basic program in source/Service-Programs called, the current program file name and the selection/cursor will be send like a RUN command.

With Android versions < 6 you have to put the selection into the clipboard, if the selection dialog hides the menu.

Unfortunately does some devices not support hardware keyboard function keys.

In this case, you can use the menu key and the up and down keys as well as the enter key.

Some devices have problems with repeated pressing of the buttons. Try another keyboard or / and an active (power supply) USB hub. Maybe you can insert batteries into your keyboard.

A keyboard or mouse with a Back key is recommended.

Navigating the program code / document

To	Press	
Select text	Shift + LEFT/RIGHT ARROW or Shift + UP/DOWN ARROW	
Move cursor right by one word	Ctrl + RIGHT ARROW	
Move cursor left by one word	Ctrl + LEFT ARROW	
Move cursor to beginning of the document	Alt+ UP ARROW	
Move cursor to end of the document	Alt + DOWN ARROW	
Move cursor to the beginning of current line	Alt + LEFT ARROW	
Move cursor to the end of the current line	Alt + RIGHT ARROW	
Select word to the left	Shift + Ctrl + LEFT ARROW	
Select word to the right	Shift + Ctrl + RIGHT ARROW	
Select from current position to beginning of the document	Shift + Alt + UP ARROW	
Select from current position to end of the document	Shift + Alt + DOWN ARROW	
Select from current position to beginning of the line	Shift + Alt + LEFT ARROW	
Select from current position to end of the line	Shift + Alt + RIGHT ARROW	

Editing (and formatting) the program code / document		
<b>To</b>	<b>Press</b>	
Undo the last action	Ctrl + Z	Not supported in the Editor
Repeat the last action	Ctrl + Y	Not supported in the Editor
Cut selected content	Ctrl + X	
Copy selected content	Ctrl + C	
Paste copied or cut content	Ctrl + V	
Select all	Ctrl + A	
Bold selected content	Ctrl + B	Not supported in the Editor
Italicize selected content	Ctrl + I	Not supported in the Editor
Underline selected content	Ctrl + U	Not supported in the Editor
Text to speech commands		
<b>To</b>	<b>Press</b>	
TextToSpeech Selection	Ctrl + 0	Starts at cursor position to the next line end or the whole selection
TextToSpeech Clipboard	Alt + Shift + 0	
TextToSpeech Stop	Alt + Ctrl + 0	



## Working with Service Programs

With the SubMenu, Hot Keys and FunctionKeys from above you are able to load and execute Service Programs coded in Basic!.

If a Sub Menu item ends with ".bas" it is linked to an equal named Basic! program in the source sub folder Service\_Programs.

If a Basic program in source/Service-Programs is called, the current program file name and the selection/cursor will be send like the RUN command.

The Service\_Programs directory is direct behind the source directory.

. . . . . /rfo-basic/source/Service Programs/

The structure is:

```
Service Programs
  CodeService.bas
  cs_data
    data
  Manual_Help.bas
  mh_data
    data
  . . .
  . . .
```

See also RUN

Example:

The blue line is inserted like the RUN command. The path part "/storage/emulated/0" is device depended. The variable ##\$ is used by the function GetBasObjectValues().

##\$="/storage/emulated/0/rfo-basic/source/myProgramName.bas?start=0?end=0?  
package=com.rfo.basicOli"

REM Example for CodeService.bas

```
FN.DEF ReLaunch(basEngine$, basProgramPath$, mode, mStart, mEnd)
eMode$ = ""
IF mode > 0
  eMode$ = "_Editor"
  IF mStart > -1 & mEnd > -1 THEN eMode$ = eMode$ + "?start=" + INT$(mStart) + "?end=" +
INT$(mEnd)
ENDIF
LIST.CREATE S, commandListPointer
LIST.ADD commandListPointer~
"new Intent(Intent.ACTION_MAIN);" ~
"setData("+ CHR$(34) + basProgramPath$ + CHR$(34) + ");" ~
"new ComponentName("+ CHR$(34) + basEngine$ + CHR$(34) + "," + CHR$(34) + basEngine$ +
".Basic" + CHR$(34) + ");" ~
"addCategory(Intent.CATEGORY_DEFAULT);" ~
"putExtra("+ CHR$(34) + "_BASIC!" + CHR$(34) + "," + CHR$(34) + eMode$ + CHR$(34) + ");" ~
%Starts program in Editor mode, if eMode$ = "_Editor"!
"addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);" ~
"addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);" ~
"addFlags(Intent.FLAG_ACTIVITY_MULTIPLE_TASK);" ~
```

```

"addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK);" ~
"EOCL"
BUNDLE.PL appVarPointer, "_CommandList", commandListPointer
APP.SAR appVarPointer
FN.END
FN.DEF InitReLaunch(objectValues)
BUNDLE.GET objectValues, "basProgramPath", basProgramPath$
BUNDLE.GET objectValues, "startSelection", startSelection
BUNDLE.GET objectValues, "endSelection", endSelection
! basEngine$ ="com.rfo.basicOli" % Or your favorite BASIC! Engine
BUNDLE.GET objectValues, "packageID", basEngine$
mode = 1
CALL ReLaunch(basEngine$, basProgramPath$, mode, startSelection, endSelection)
FN.END
FN.DEF GetBasObjectValues (objectValues)
GLOBALS.FNIMP $$$ % On start a line started with $$$ is added by Basic!. See also the RUN
command.
SPLIT.ALL spRes$[], $$$, "\\?" % ? is a control delimiter, \\ Regular Exp. specific, because ? sign
ARRAY.LENGTH al, spRes$[]
IF al > 0
BUNDLE.PUT objectValues, "basProgramPath", spRes$[1]
BUNDLE.PUT objectValues, "startSelection", VAL(MID$(spRes$[2], 7))
BUNDLE.PUT objectValues, "endSelection", VAL(MID$(spRes$[3], 5))
BUNDLE.PUT objectValues, "packageID", VAL(MID$(spRes$[4], 9))
ENDIF
FN.END

!*** Main Part ***
BUNDLE.CREATE objectValues
GetBasObjectValues (objectValues)

!* The Selection Parameters
!* If startSelection = endSelection THEN it is the cursor position
!* The position values start with 0 (before the first sign)
BUNDLE.GET objectValues, "startSelection", startSelection
BUNDLE.GET objectValues, "endSelection", endSelection
? startSelection
? endSelection

CLIPBOARD.GET mClipData$ % Get automatically created text from the selected Basic source
contents
? mClipData$

!* Insert your code here!
sel = -6000
DIALOG.MESSAGE "Do your code here!", "In 6 seconds we will return!", sel

CLIPBOARD.PUT mClipData$

!* Set the new cursor/selection position(s)
BUNDLE.PUT objectValues, "startSelection", startSelection
BUNDLE.PUT objectValues, "endSelection", endSelection

InitReLaunch(objectValues)
EXIT

```

## Color Table

Predefined Colors based on official **HTML Color Names** of the World Wide Web Consortium (W3C)

Also possible "`_{<alpha>},HSV<hue[0...360]>{,<saturation [0...1]>}, <valueOfBrightness [0...1]>}`". The string "`_200,HSV300`" returns the color blue with alpha = 200.

HTML Color Name	Basic! Color Code	Basic! Color Name	HTML Color Code Hex
AliceBlue	<b>255</b> ,240,248,255	<b>_255</b> ,AliceBlue	#FFF0F8FF
AliceBlue	240,248,255	_AliceBlue	#F0F8FF
AntiqueWhite	250,235,215	_AntiqueWhite	#FAEBD7
Aqua	0,255,255	_Aqua	#00FFFF
Aquamarine	127,255,212	_Aquamarine	#7FFFD4
Azure	240,255,255	_Azure	#F0FFFF
Beige	245,245,220	_Beige	#F5F5DC
Bisque	255,228,196	_Bisque	#FFE4C4
Black	0,0,0	_Black	#000000
			or #000
			or #F000
BlanchedAlmond	255,235,205	_BlanchedAlmond	#FFEBCD
Blue	0,0,255	_Blue	#0000FF
BlueViolet	138,43,226	_BlueViolet	#8A2BE2
Brown	165,42,42	_Brown	#A52A2A
BurlyWood	222,184,135	_BurlyWood	#DEB887
CadetBlue	95,158,160	_CadetBlue	#5F9EA0
Chartreuse	127,255,0	_Chartreuse	#7FFF00
Chocolate	210,105,30	_Chocolate	#D2691E
Coral	255,127,80	_Coral	#FF7F50
CornflowerBlue	100,149,237	_CornflowerBlue	#6495ED
Cornsilk	255,248,220	_Cornsilk	#FFF8DC
Crimson	220,20,60	_Crimson	#DC143C
Cyan	0,255,255	_Cyan	#00FFFF
DarkBlue	0,0,139	_DarkBlue	#00008B
DarkCyan	0,139,139	_DarkCyan	#008B8B
DarkGoldenRod	184,134,11	_DarkGoldenRod	#B8860B
DarkGray	169,169,169	_DarkGray	#A9A9A9
DarkGreen	0,100,0	_DarkGreen	#006400
DarkKhaki	189,183,107	_DarkKhaki	#BDB76B
DarkMagenta	139,0,139	_DarkMagenta	#8B008B
DarkOliveGreen	85,107,47	_DarkOliveGreen	#556B2F
DarkOrange	255,140,0	_DarkOrange	#FF8C00
DarkOrchid	153,50,204	_DarkOrchid	#9932CC
DarkRed	139,0,0	_DarkRed	#8B0000
DarkSalmon	233,150,122	_DarkSalmon	#E9967A
DarkSeaGreen	143,188,143	_DarkSeaGreen	#8FBC8F
DarkSlateBlue	72,61,139	_DarkSlateBlue	#483D8B
DarkSlateGray	47,79,79	_DarkSlateGray	#2F4F4F
DarkTurquoise	0,206,209	_DarkTurquoise	#00CED1
DarkViolet	148,0,211	_DarkViolet	#9400D3
DeepPink	255,20,147	_DeepPink	#FF1493
DeepSkyBlue	0,191,255	_DeepSkyBlue	#00BFFF
DimGray	105,105,105	_DimGray	#696969
DodgerBlue	30,144,255	_DodgerBlue	#1E90FF
FireBrick	178,34,34	_FireBrick	#B22222

HTML Color Name	Basic! Color Code	Basic! Color Name	HTML Color Code Hex
FloralWhite	255,250,240	_FloralWhite	#FFFAF0
ForestGreen	34,139,34	_ForestGreen	#228B22
Fuchsia	255,0,255	_Fuchsia	#FF00FF
Gainsboro	220,220,220	_Gainsboro	#DCDCDC
GhostWhite	248,248,255	_GhostWhite	#F8F8FF
Gold	255,215,0	_Gold	#FFD700
GoldenRod	218,165,32	_GoldenRod	#DAA520
Gray	128,128,128	_Gray	#808080
Green	0,128,0	_Green	#008000
GreenYellow	173,255,47	_GreenYellow	#ADFF2F
HoneyDew	240,255,240	_HoneyDew	#F0FFF0
HotPink	255,105,180	_HotPink	#FF69B4
IndianRed	205,92,92	_IndianRed	#CD5C5C
Indigo	75,0,130	_Indigo	#4B0082
Ivory	255,255,240	_Ivory	#FFFFFF
Khaki	240,230,140	_Khaki	#F0E68C
Lavender	230,230,250	_Lavender	#E6E6FA
LavenderBlush	255,240,245	_LavenderBlush	#FFF0F5
LawnGreen	124,252,0	_LawnGreen	#7CFC00
LemonChiffon	255,250,205	_LemonChiffon	#FFFACD
LightBlue	173,216,230	_LightBlue	#ADD8E6
LightCoral	240,128,128	_LightCoral	#F08080
LightCyan	224,255,255	_LightCyan	#E0FFFF
LightGoldenRodYellow	250,250,210	_LightGoldenRodYellow	#FAFAD2
LightGray	211,211,211	_LightGray	#D3D3D3
LightGreen	144,238,144	_LightGreen	#90EE90
LightPink	255,182,193	_LightPink	#FFB6C1
LightSalmon	255,160,122	_LightSalmon	#FFA07A
LightSeaGreen	32,178,170	_LightSeaGreen	#20B2AA
LightSkyBlue	135,206,250	_LightSkyBlue	#87CEFA
LightSlateGray	119,136,153	_LightSlateGray	#778899
LightSteelBlue	176,196,222	_LightSteelBlue	#B0C4DE
LightYellow	255,255,224	_LightYellow	#FFFFE0
Lime	0,255,0	_Lime	#00FF00
LimeGreen	50,205,50	_LimeGreen	#32CD32
Linen	250,240,230	_Linen	#FAF0E6
Magenta	255,0,255	_Magenta	#FF00FF
Maroon	128,0,0	_Maroon	#800000
MediumAquaMarine	102,205,170	_MediumAquaMarine	#66CDAA
MediumBlue	0,0,205	_MediumBlue	#0000CD
MediumOrchid	186,85,211	_MediumOrchid	#BA55D3
MediumPurple	147,112,219	_MediumPurple	#9370DB
MediumSeaGreen	60,179,113	_MediumSeaGreen	#3CB371
MediumSlateBlue	123,104,238	_MediumSlateBlue	#7B68EE
MediumSpringGreen	0,250,154	_MediumSpringGreen	#00FA9A
MediumTurquoise	72,209,204	_MediumTurquoise	#48D1CC
MediumVioletRed	199,21,133	_MediumVioletRed	#C71585
MidnightBlue	25,25,112	_MidnightBlue	#191970
MintCream	245,255,250	_MintCream	#F5FFFA
MistyRose	255,228,225	_MistyRose	#FFE4E1
Moccasin	255,228,181	_Moccasin	#FFE4B5
NavajoWhite	255,222,173	_NavajoWhite	#FFDEAD
Navy	0,0,128	_Navy	#000080
OldLace	253,245,230	_OldLace	#FDF5E6
Olive	128,128,0	_Olive	#808000

HTML Color Name	Basic! Color Code	Basic! Color Name	HTML Color Code Hex
OliveDrab	107,142,35	_OliveDrab	#6B8E23
Orange	255,165,0	_Orange	#FFA500
OrangeRed	255,69,0	_OrangeRed	#FF4500
Orchid	218,112,214	_Orchid	#DA70D6
PaleGoldenRod	238,232,170	_PaleGoldenRod	#EEE8AA
PaleGreen	152,251,152	_PaleGreen	#98FB98
PaleTurquoise	175,238,238	_PaleTurquoise	#AFEEEE
PaleVioletRed	219,112,147	_PaleVioletRed	#DB7093
PapayaWhip	255,239,213	_PapayaWhip	#FFEFD5
PeachPuff	255,218,185	_PeachPuff	#FFDAB9
Peru	205,133,63	_Peru	#CD853F
Pink	255,192,203	_Pink	#FFC0CB
Plum	221,160,221	_Plum	#DDA0DD
PowderBlue	176,224,230	_PowderBlue	#B0E0E6
Purple	128,0,128	_Purple	#800080
RebeccaPurple	102,51,153	_RebeccaPurple	#663399
Red	255,0,0	_Red	#FF0000
RosyBrown	188,143,143	_RosyBrown	#BC8F8F
RoyalBlue	65,105,225	_RoyalBlue	#4169E1
SaddleBrown	139,69,19	_SaddleBrown	#8B4513
Salmon	250,128,114	_Salmon	#FA8072
SandyBrown	244,164,96	_SandyBrown	#F4A460
SeaGreen	46,139,87	_SeaGreen	#2E8B57
SeaShell	255,245,238	_SeaShell	#FFF5EE
Sienna	160,82,45	_Sienna	#A0522D
Silver	192,192,192	_Silver	#C0C0C0
SkyBlue	135,206,235	_SkyBlue	#87CEEB
SlateBlue	106,90,205	_SlateBlue	#6A5ACD
SlateGrey	112,128,144	_SlateGrey	#708090
Snow	255,250,250	_Snow	#FFFAFA
SpringGreen	0,255,127	_SpringGreen	#00FF7F
SteelBlue	70,130,180	_SteelBlue	#4682B4
Tan	210,180,140	_Tan	#D2B48C
Teal	0,128,128	_Teal	#008080
Thistle	216,191,216	_Thistle	#D8BFD8
Tomato	255,99,71	_Tomato	#FF6347
Turquoise	64,224,208	_Turquoise	#40E0D0
Violet	238,130,238	_Violet	#EE82EE
Wheat	245,222,179	_Wheat	#F5DEB3
White	255,255,255	_White	#FFFFFF
WhiteSmoke	245,245,245	_WhiteSmoke	#F5F5F5
Yellow	255,255,0	_Yellow	#FFFF00
YellowGreen	154,205,50	_YellowGreen	#9ACD32