STRINGPLAYER

User Guide Application Version 3.0.0

INTRODUCTION

StringPlayer is a Laird developed utility for Windows Platform which is used to expedite regression testing of any UART-based module and in future TCP/IP based testing will also be possible.

It has a scripting engine that allows scripts written in the language defined in this user manual to be compiled and run.

In general a script will consist of many tests, each beginning with the TITLE statement.

If a test fails, then it will try MAXTRY times (specified via command line) and if it still fails that many times, the scripting engine will automatically progress to the next test which is identified by the next TITLE statement. On retry, the scripting runtime engine will reset its program counter to the most recent TITLE statement.

Arithmetic and conditional looping is possible. So is the ability to launch other windows applications from within the script and optionally wait for those applications to close.

User interaction via dialog boxes can be done.

SCRIPT SYNTAX DESCRIPTION

- Blank line are ignored
- Each valid line is a statement or a directive
- Directive lines start with a # character
- Tokens are case insensitive
- Statements and Directives cannot span multiple lines
- Tokens are separated by spaces
- String parameters can embed nonprintable characters using the escape sequence \HH where HH is the ASCII hex value.

  Special cases are:

  ```
  -  '\r'  == carriage return    0x0D
  -  '\n'  == line feed          0x0A
  -  '\b'  == backspace          0x08
  -  '\41' == A
  ```

- String parameters can embed variables using the syntax [varname]
  - To embed the [character enter the three character sequence \5B
- Constant strings can be hexized using the syntax [%ABCD] where the 4 character string 'ABCD' is replaced by the 8 character string '41424344'. Particularly useful when a port is opened using PORT_MPOINT or PORT_BINHEX
- Pre-declared system variables are available

GENERAL

StringPlayer was formerly known as *AtProtocolTest.exe*. It is capable of testing text and binary protocols over multiple COM ports and/or pipes. For binary protocols, the binary data is sent, received, and displayed as hex strings.

A HTML log file containing everything is created. In addition, StringPlayer places failures in a separate HTML file and generates a text file which lists the failed tests.

COMMAND LINE ARGUMENTS

The utility takes the following case-insensitive command line arguments:

Command Line Argument	Definition
TRACE	Used for debugging
AGGLOG	Displays the aggregate comms activity screen instead of separate screens for individual logical streaming channels.
AGGLOGTIMESTAMP	All activity in the aggregate comms activity screen comms window has a timestamp
QUIT	The application terminates upon completion.
ARG=XXX	Multiples of these can be specified. XXX is stored in an internal array. After each ARG=XXX, a counter is incremented to allow the next XXX to be stored in the array of strings. The text substitution results from the script that runs the variable enclosed in [] (as per [SysVarArgX]). See **SysVarArg0**.
name:value	Multiple of these can be specified. Each one will result in a StringPlayer variable called 'name' and will be initialised with 'value'. If 'value' has whitespace then enclose it all in double quotes like "name:value". In the stringplayer script you can reference the value using the usual [name] syntax
SCRIPT = SomeFileName	The specified script is processed. If a script is not specified, then on startup, an open file dialog box allows you to pick a script for processing.
CWD = SomeFolderName	When the application starts, *SomeFolderName* is set as the current working dictionary.
MAXTRY=NNN	The decimal number (NNN) is clipped between 1 and 100. This indicates the number of times it tries a test (which began from the most recent TITLE statement).
LOGPATH=SomeFolderName	Indicates the absolute or relative path for the folder where the logfile is saved.
MAXLINES=N	Reduces the main window display to display up-to the most recent N lines and clear previous data (useful for huge automated testing scripts). Minimum N is 100 and maximum is 10000. If not specified then N=750 will be assumed. Note that this only alters the display and all data is included in the log files regardless of this setting.
SMALLLOG	Deprecated. Use MAXLINES=N instead.

Command Line Argument	Definition
	If used then it is equivalent to MAXLINES=500
SKIP	Allows skipping individual tests by clicking the 'Skip' button on the main window.
MAXPORTS=N	Change maximum logical ports to N where N is 16 to 255. If N is <16 then it defaults to 16 and if >255 it clips to 255.

PREDEFINED SYSTEM VARIABLES

System Variable	Definition
SysVarAppVersion	Contains the test application version string as shown in the title bar.
SysVarArg0 .. SysVarArg15	Contains the value of ARG=something command line arguments when the application is invoked. The first ARG=XX goes into SysVarArg0 and so on.
SysVarMaxPorts	Contains the maximum number of serial ports that can be managed.
SysVarMatchCount	When the statement CONTAINS is processed, this variable is updated with the number of times the match was encountered.
SysVarFailOnVarMissing	If this string has a non-zero integer value then, when a variable is encountered and it is missing, the script fails.
SysVarMagicCmd	Contains the magic command for invoking factory default mode on power up.
SysVarOnFailMessage	Contains a null string by default. This indicates that the default message is printed when a test fails. Otherwise, the content of this message is printed.
SysVarMsgBoxBtnVal	When the blocking statement MESSAGEBOX returns because the user selected a button, this variable contains a decimal string with value in the range 1 to 7. This signifies the button that was clicked as follows: 1. `OK` 2. `CANCEL` 3. `ABORT` 4. `RETRY` 5. `IGNORE` 6. `YES` 7. `NO`
SysVarMsgBoxBtnName	When the blocking statement MESSAGEBOX returns because the user selected a button, this variable contains a descriptive string which signifies that the button was clicked as follows: ▪ `ok` ▪ `cancel` ▪ `abort` ▪ `retry` ▪ `ignore` ▪ `yes` ▪ `no` ▪ `unknown` (contact Laird if this occurs)

DIRECTIVES

#INCLUDE *filename*
Include and compile the script defined in filename.

#INCLUDEFOLDERn *foldername*
foldername is used to look for #include files if the file is not found in the script folder.

- Up to version 1.6.x – n was in the range 0 to 3 (inclusive)
- As of version 1.7.0 – n is in the range 0 to 9 (inclusive)

#EXECUTE *"appName optional parameters" "default_folder_for_child_app"*
Runs the application *appName* at compile time and immediately proceeds to the next statement. The child process window displays. Any portion of the string parameter can be runtime modified using the [] technique.

#EXECUTEWAIT *"appName optional parameters" "default_folder_for_child_app"*
Runs the application *appName* at compile time and waits until it exits. The child process window displays. Any portion of the string parameter can be runtime modified using the [] technique.

#EXECUTEHIDE *"appName optional parameters" "default_folder_for_child_app"*
Runs the application *appName* at compile time and immediately proceeds to the next statement. The child process window is hidden. Any portion of the string parameter can be runtime modified using the [] technique.

#EXECUTEWAITHIDE *"appName optional parameters" "default_folder_for_child_app"*
Runs the application appName at compile time and waits until it exits. The child process window is hidden. Any portion of the string parameter can be runtime modified using the [] technique.

STATEMENTS

The following are arithmetic commands.

<ADD|MUL|DIV|SUB|TST|MOD|AND|OR|XOR|ROR|ROL|ABS> <varname> <value>
Performs the arithmetic using "value" converted to an integer to varname and result is stored back as a string – except for TST which is as per SUB but the variable is not updated. An internal ConditionFlag register is updated so that it can be used with the IF<cond> and JUMP<cond> commands.

<HEXADD|HEXMUL|HEXDIV|HEXSUB|HEXTST|HEXMOD > <varname> <hexvaluestring>
<HEXAND|HEXOR|HEXXOR|HEXROR|HEXROL> <varname> <value>
Performs the hex arithmetic using " hexvaluestring " converted to an integer to varname and the result is stored back as a string – except for TST which is as per HEXSUB but the variable is not updated. An internal ConditionFlag register is updated appropriately so that it can be used with the IF<cond> and JUMP<cond> commands.

BREAKOFF <portnum> <timeoutMs>
Release the BREAK on Logical pipe <portnum> and wait for 'timeoutms' milliseconds

BREAKON <portnum> <timeoutMs>
Send a BREAK out from Logical pipe <portnum> for 'timeoutms' milliseconds

CHECK <varname> <string>
Fails the test if value "string" does not match content of 'varname'

CHECKCTS <port> "<ON | OFF>"
Read the modem CTS status for serial port 'port'. The state can be either ON or OFF (in caps). ZERO flag is set to 1 if the state matches specified value, otherwise 0

CHECKDCD <port> "<ON | OFF>"
Read the modem DCD status for serial port 'port'. The state can be either ON or OFF (in caps). ZERO flag is set to 1 if the state matches specified value, otherwise 0

CHECKDSR <port> "<ON | OFF>"
Read the modem DSR status for serial port 'port'. The state can be either ON or OFF (in caps). ZERO flag is set to 1 if the state matches specified value, otherwise 0

CHECKRI <port> "<ON | OFF>"
Read the modem RI status for serial port 'port'. The state can be either ON or OFF (in caps). ZERO flag is set to 1 if the state matches specified value, otherwise 0

CONSOLEUPDATE <true/false>
By default this is true on start so that the main window displays messages as the script runs. Setting to false will disable that which will help prevent false failures because StringPlayer is not keeping up with all the data comms activity.

CONST <varname> <string>
Declare variable 'varname' and store value "string" without the quotes in 'varname'. "string" can have embedded [cName] so that it is a combination of other earlier defined constants.
For example:
```
CONST cName0 "hello"
CONST cName1 "[cName0] world"
```

CONTAIN <varname> <string>
Fails the test if value "string" is not present as a substring in the content of 'varname'. The system variable SysVarMatchCount is updated.

DEC2HEX <varname> <decimal_integer_value_string>
Convert the decimal string into an integer string and then store the hexadecimal equivalent in the variable. See also counterpart HEX2DEC

DECLARE <varname>
This declares a variable with name 'varname'

DEFTIMEOUT <timeoutMs>
Set default timeout in milliseconds. The parameter is in quotes so that variables can be embedded which will be parsed at runtime.

DELAY <timeoutMs>
Wait for 'timeoutms' milliseconds before next action. The parameter is in quotes so that variables can be embedded which will be parsed at runtime.
The time scalar specified by the TIMESCALAR statement is applied to <timeoutMs>, so for example if the most recent TIMESCALAR specified 300 (as in percent) then the actual delay will be 300% of the value specified in <timeoutMs>

DELAYABS <timeoutMs>
Wait for 'timeoutms' milliseconds before next action. The parameter is in quotes so that variables can be embedded which will be parsed at runtime.
The time scaler specified by the TIMESCALAR statement is not applied to <timeoutMs>

END
This defines the end of the script at runtime and shows the pass/fail statistics

EXIT
This defines the end of the script at runtime and does not show any additional information.

EXECUTE <appName and [optional parameters]>
Runs the application with parameters and will NOT block. Any portion of the string parameter can be runtime modified using the [] technique.

EXECUTEEX <nFlags> <appName and [optional parameters]>
Runs the application with parameters and will NOT block. Any portion of the string parameter can be runtime modified using the [] technique. nFlags is a bit mask.
0 : Set to hide the executed app
1 : Set to minimize the executed app

EXECUTEWAIT <appName and [optional parameters]>
Runs the application with parameters and will block until the application 'appName' exits. Any portion of the string parameter can be runtime modified using the [] technique.

EXECUTEWAITRET <appName and [optional parameters]> <varExitCode>
Runs the application with parameters and will block until the application 'appName' exits, returning the application exit code into the supplied variable. **Any portion of the string parameter can be runtime modified using the [] technique.**

EXECUTEWAITEX <nFlags> <appName and [optional parameters]> <optional_varname>
Runs the application with parameters and will block. Any portion of the string parameter can be runtime modified using the [] technique. nFlags is a bit mask.
0 : Set to hide the executed app
1 : Set to minimize the executed app
If <optional_varname> is provided, then it will contain the exitvalue of the application

FAIL [message]
Fails the script, so that the program counter skips to the next TITLE action.

FILECOMPARE <true | false> <filenameA> <filenameB>
Fails the test if "filenameA" does not match content of "filenameB" if true is specified. If false is specified then fails the test if files match.

FILEEXISTS <true | false> <filename>
Fails the test if "filename" does not exist and true is specified

FILEOPEN <handle> <READ|WRITE|APPEND> <filename>
Open "filename" for reading or writing (or appending)

FILECLOSE <handle>
Close file associated with handle

FILEREAD <handle> <varData> <varLen>
Read varLen bytes of text into varData. varLen is updated with actual number of bytes read

FILTERMOVE <NOT> <bitmask> <varDst> <varSrc>
The NOT is optional.
Move characters from <varSrc> to <varDst> as long as the character satisfies the filter specified in <bitmask>. Stop when a character is encountered that does not satisfy any condition in the mask.
For example bitmask=0 will result in an empty varDst.
Bitmask is as follows and bit 0 is checked first, then bit 1 etc :-

```
0x000001 octal digits
0x000002 decimal digits
0x000004 isxdigit
0x000008 'A' to 'Z'
0x000010 'a' to 'z'
0x000020 ' '
0x000040 '.'
0x000080 ','
0x000100 '('
0x000200 ')'
0x000400 '{'
0x000800 '}'
0x001000 '['
0x002000 ']'
0x004000 ispunct
0x008000  isspace
0x010000  iscntrl
0x800000 >=0x80
```

If the NOT argument is supplied then it implies that f the character matches then it will **end** the move process – which is logic inversion.

FILEWRITE <handle> <varData>
Write varData to file associated with handle

FOLDEREXISTS <true | false> <foldername>
Fails the test if "foldername" does not exist and true is specified

GETCWD <varname>
Gets the current working folder. That is, where the exe was launched from.

HEX2DEC <varname> <hexadecimal_integer_value_string>
Convert the hex string into an integer string and then store the decimal equivalent in the variable. See also counterpart DEC2HEX

IF<cond>
<cond> is one of <Z|NZ|LT|LE|GT|GE> and if the appropriate flag in the conditionflag register is true then script actions the next item otherwise skips to the one beyond it
Where:

- Z = Zero
- NZ = NonZero
- LT = LessThan
- LE = LessThanOrEqual
- GT = GreaterThan
- GE = GreaterThanOrEqual

INRANGE <percentage> <decimal_target_value_string> <decimal_value_string>

Checks that <decimal_value_string> is withing +/- <percentage> of <decimal_target_value_string>.
If not then the test fails.

JUMP <labelname>

Unconditionally jump to labelname

JUMP <cond> <labelname>

<cond> is one of <Z|NZ|LT|LE|GT|GE> and if the appropriate flag in the conditionflag register is true then script program counter jumps to 'labelname'
Where:

- Z = Zero
- NZ = NonZero
- LT = LessThan
- LE = LessThanOrEqual
- GT = GreaterThan
- GE = GreaterThanOrEqual

FLUSHRX <portnum>

Empty the receive buffer of Logical pipe <portnum>

FLUSHTX <portnum>

Empty the transmit buffer of Logical pipe <portnum>

LABEL <labelname>

Marks the action as a label. "labelname" only has a script file scope.

MATCHCOUNT <varname> <string>

Counts the number of times “string” appears in varname. The system variable SysVarMatchCount is updated which can then be used with the arithmetic functions.

MESSAGEBOX <messagetext> <dialogcaption> <buttons>

This is a blocking statement which is used to display a modal dialog box with text “messagetext” and a dialog box title “dialogcaption” with 1 or more buttons. The buttons displayed are dependent on the value of the integer value ‘buttons’ as follows:

0	Yes, No
1	Yes, No, Cancel
2	Ok
3	Ok, Cancel
4	Retry, Cancel
5	Abort, Retry, Ignore
Other	Abort, Retry, Ignore

When the user selects a button, the system variable SysVarMsgBoxBtnVal will contain a decimal value string in the range 1 to 7 inclusive as follows:

1	OK
2	CANCEL
3	ABORT
4	RETRY
5	IGNORE
6	YES
7	NO

If you prefer a descriptive text describing the button that was pressed, then use the system variable SysVarMsgBoxBtnName

MID$ varname *string* Offset Length

The substring of “string” starting at Offset (0 indexed, but can be negative) and Length is copied into the varname. If Offset is negative, then the right most substring is copied. For example, -4 4 will copy the last 4 characters of the source string.

MULTIINDEX <varname> <portnum>

Returns the most recent matching index string number after WAITMULTI was processed. The return value is a 0 indexed value.

NOTCONTAIN <varname> <string>

Fails the test if value "string" is present as a substring in the content of 'varname'. The system variable SysVarMatchCount is updated.

PKTCOUNTGET <portnum> <varname>

Get the packet count and save in variable

PKTCOUNTRESP <portnum> <varname>

Get the packet count in the Rx buffer and save in variable

PKTCOUNTSET <portnum> <varname>

Set the packet count to the value specified

PRINT <string>

Display message string "string" in green color in the output screen. Non-printable characters are not translated into '\hh'.

PRINTE <string>

Display message string "string" in green color in the output screen. Non-printable characters are translated into '\hh'.

PORT <portnum> <comport> <string>

Associate 'comport' which is in format COMnn with portnum.
<string> is for example “9600,N,8,1 [,0|1] [,0|1][,nnn][,mmm]”
The port is opened. If it fails to open then script run is aborted The first <,0|1> in the string is optional and specifies CTS/RTS handshaking. If it is not specified, then CTS/RTS is enabled.
The second <,0|1> in the string is optional and specifies modem status change detection. If it is not specified, then it is enabled. It is advised to set this to 0 if the device will thrash the PC’s CTS input signal and an exception “No more trigger slots” is displayed.
The [,nnn] parameter is an integer value in range 0 to 5000 and it specifies the time to wait for port to stabilise after opening it in milliseconds. If not specified then default will be 0.
The [,mmm] parameter is an integer value in range 0 to 5000 and it specifies the time to wait for port to stabilise after closing it in milliseconds. If not specified then default will be 0.
If on optional paramenter is missing then it should be entered as ,,

PORT_CLOSE <portnum>

Close the logical portnum specified

PORT_HEXBIN <portnum> <comport> <string>

Associate 'comport' which is in format COMn with portnum.
<string> is for example “9600,N,8,1 [,0|1] [,0|1] [,nnn][,mmm]”
The port is opened. If it fails to open then script run is aborted.
The communications is assumed to binary so at script level the Rx/Tx data is as hex digits.
Which means in the script you send strings which contain hex digits only and wait on strings that are also hex digits only.
The first <,0|1> in the string is optional and controls CTS/RTS handshaking. If unspecified, CTS/RTS is enabled.
The second <,0|1> in the string is optional and controls modem status change detection. If unspecified, then it is enabled. Laird advises you set this to 0 if the device will thrash the PC’s CTS input signal and an exception “No more trigger slots” is displayed.
The [,nnn] parameter is an integer value in range 0 to 5000 and it specifies the time to wait for port to stabilise after opening it in milliseconds. If not specified then default will be 0.
The [,mmm] parameter is an integer value in range 0 to 5000 and it specifies the time to wait for port to stabilise after closing it in milliseconds. If not specified then default will be 0.
If on optional paramenter is missing then it should be entered as ,,

PORT_MPOINT <portnum> <comport> <string>

Associate 'comport' which is in format COMn with portnum.
<string> is for example “9600,N,8,1 [,0|1] [,0|1] [,nnn][,mmm]”
The port is opened. If it fails to open then script run is aborted.
The communications is assumed to binary so at script level the Rx/Tx data is as hex digits and the protocol is assumed to be the BISM multipoint.
The first <,0|1> in the string is optional and controls CTS/RTS handshaking. If unspecified, CTS/RTS is enabled.
The second <,0|1> in the string is optional and controls modem status change detection. If unspecified, then it is enabled. Laird advises you set this to 0 if the device will thrash the PC’s CTS input signal and an exception “No more trigger slots” is displayed.
The [,nnn] parameter is an integer value in range 0 to 5000 and it specifies the time to wait for port to stabilise after opening it in milliseconds. If not specified then default will be 0.
The [,mmm] parameter is an integer value in range 0 to 5000 and it specifies the time to wait for port to stabilise after closing it in milliseconds. If not specified then default will be 0.
If on optional paramenter is missing then it should be entered as ,,

PORT_PIPE <portnum> “<appname_command_arguments>” “<current_folder_for_the launched_app>”
Opens a pipe to the executable file ‘appname and command line arguments’ on logical port ‘portnum’.
Then launched application will inherit the current folder as its default current folder

PORTINFO <varname> <portnum> <nInfoId>

Returns port info identified by nInfoId in the variable specified, where nInfoId is:

nInfoId	Description
0	Bytes received since last time PORTINFO called with nInfoId = 0. * The count is reset as a result of the read.
1	Bytes received since last time PORTINFO was called with nInfoId = 0 * The count is not reset.
2	Total number of bytes in underlying rx buffer (ATHOST class only)
16	Free space in transmit buffer
32	Number of times CTS has toggled
33	Number of times DSR has toggled
34	Number of times DCD has toggled
35	Number of times RI has toggled

RXCOUNT

This statement does not exist, but it is mentioned here to help you go to the function that will provide the information SEE PORTINFO with nInfoId set to 2

SAVERESP <varname> <portnum>

Stores the current content of the receive buffer in Logical pipe <portnum> to the variable 'varname'. The response string is echoed to the screen

SAVERESPNE <varname> <portnum>

Stores the current content of the receive buffer in Logical pipe <portnum> to the variable 'varname'
The response string is not echoed to the screen

SEND <timeoutMs> <portnumA> <txstr> <portnumB> <rxstr>

Transmit "txstr" from COM<portnumA> and wait for "rxstr" to arrive at serial port COM<portnumB>. The receive string MUST arrive within timeoutMs.
Set timeoutMs to -1 to use default timeout which is set via command DEFTIMEOUT described above.

Note:The timeout value is scaled according to the percent value specified by the most recent TIMESCALAR statement

SENDCMD <portnum> <txstr>

Transmit "txstr" from COM<portnum> after flushing the receive buffer. Proceed immediately to next action

Note:The timeout value is scaled according to the percent value specified by the most recent TIMESCALAR statement

SENDMPDATA <portnum> <channel> <txstr>

Transmit "txstr" from COM<portnum> in a data packet in logical channel specified. Only data is supplied in “txstr”, the packet length and channel numbers are automatically prepended.

Note:The timeout value is scaled according to the percent value specified by the most recent TIMESCALAR statement

SENDWAIT <timeoutMs> <portnum> <txstr>

Transmit "txstr" from Logical pipe <portnum> and wait for timeoutMs

Note:The timeout value is scaled according to the percent value specified by the most recent TIMESCALAR statement

SENDWAITOK <timeoutMs> <portnum> <txstr>

Transmit "txstr" from Logical pipe <portnum> and wait for an "\r\nOK\r\n"

Note:The timeout value is scaled according to the percent value specified by the most recent TIMESCALAR statement

SENDWAITERR <timeoutMs> <portnum> <txstr>

Transmit "txstr" from Logical pipe <portnum> and wait for an "\r\nERROR"

Note:The timeout value is scaled according to the percent value specified by the most recent TIMESCALAR statement

SET <varname> <string>

Updates variable 'varname' with the data in 'string'. It requires that 'varname' variable has been created using the DECLARE statement

SETX <varname> <string>

Updates variable 'varname' with the data in 'string'. If the variable does not exist then it will automatically be created as if DECLARE had been used.
Very useful when #including .sbr files which require parameters.

SETCOMMS <portnum> 9600,N,8,1 [,0|1] [,0|1]

Change the serial comport Logical pipe <portnum> so that it uses parameters specified.
The first <,0|1> in the string is optional and specifies CTS/RTS handshaking. If it is not specified, then CTS/RTS is enabled.
The second <,0|1> in the string is optional and specifies modem status change detection. If it is not specified, then it is enabled. It is advised to set this to 0 if the device will thrash the PC's CTS input signal and an exception "No more trigger slots" is displayed.
If the first is missing but second present then the separator should be ,,

SETDTR <port> <ON|OFF>

Sets DTR output to ON or OFF for port 'port'

SETRTS <port> <ON|OFF>

Sets RTS output to ON or OFF for port 'port'.
Handshaking has to be disabled in PORT and PORT_MPOINT for this to work

SETVAR <varname> <string>

Store value "string" without the quotes in 'varname'

SPRINTHEX <varname> <hex_integer_value_string>

On entry, varname must contain a format string compatible with the 'C' printf function (if it is empty, then %d is assumed). The "hexstring" is then converted to an integer value using the 'C' scanf() function.
Finally that integer value is fed into a 'C' sprintf() function and the output string is stored back in varname.
For example:
SET vTmp "%d"
SPRINTHEX vTmp "102"
PRINT "[vTmp]"
Results in PRINT outputting "258"

SPRINTINT <varname> <decimal_integer_value_string>

On entry, varname must contain a format string compatible with the 'C' printf function (if it is empty, then %d is assumed). The "string" is then converted to an integer value using the 'C' atoi() function.
Finally that integer value is fed into a 'C' sprintf() function and the output string is stored back in varname.
For example:

```
SET vTmp "0x%04X"
SPRINTINT vTmp "258"
PRINT "[vTmp]"
Results in PRINT outputting "0x0102"
```

STRCAT <varname1> <string>

Concatenate "string" to variable varname1

STRCMP <varname1> <varname2>

Compares the two strings and the condition flag Z is updated. Z will be set if the two compare.

STRCMPI <varname1> <varname2>

Compares the two strings ignoring case and the condition flag Z is updated. Z will be set if the two compare.

STRDEHEXIZE <flags> <varname1> <varname2>

This command will attempt to de-hexize content of varname2 and save the resulting binary string in varname1. It will abort if it encounters a 00 as that is null.
Please note: if 5B or 5D is encountered which are [and] characters respectively, then they will be added to varname1 as the two letter sequence \[and \]

STRLEN <varname1> <varname2>

Stores the length of the string in varname1 as a value string in varname2.
To be precise, varname2 will contain the length.

STRSTR <varname1> <varname2>

Check if varname2 string is contained inside varname1 string and the compare is case sensitive and the condition flag Z is updated. Z will be set if the varname2 is present in varname1.

STRSTRI <varname1> <varname2>

Check if varname2 string is contained inside varname1 string and the compare is case insensitive and the condition flag Z is updated. Z will be set if the varname2 is present in varname1.

SUBTITLE <string>

Defines a label used for marking the start of a test sub-group of actions, as part of a bigger test group defined using title.

TIMER START <varname>

Start timer (msec resolution)

TIMER ELAPSE <varname1> <varname2>

Stores the number of msec elapsed since timer varname1 was started in variable varname2

TIMER EXPIRED <varname> <string_msec>

If the time elapsed since varname timer was started using TIMER START has elapsed by more that "string_msec" then the zero flag is set so that IFNZ/JUMPNZ will succeed.

TIMESCALAR <percent_value>

Set the time scalar for all subsequent timeouts (except for DELAYABS statement) and the value is in percent, so for example 50 will halve the timeouts and 200 will double them.
It can be specified multiple times

TITLE <string>
Defines a label used for marking the start of a test group of actions. If an action fails, then all actions up to the next TEST label are skipped so that an overnight test which has some failures is not aborted totally.
As of v1.9.70 if the command line argument MAXTRY=NNN was specified when the application was launched, then the most recent TITLE statement processed will be the destination of the rewind location.

TRACE <true|false>
If true then tracing will be enabled. When false, tracing will not be enabled.

TRACEWIN <true|false>
If true then the aggregate log window is updated. When false the aggregate window is displayed but not updated.

TXSTRING <portnumA> <txstr>
Transmit "txstr" from COM<portnumA>. Proceed immediately to next action

TXSTRINGEX <portnumA> <txstr> <repeatCount>
Transmit "txstr" from COM<portnumA> repeatCount times.
Proceed immediately to next action

TXUNICODE <portnumA> <txstr>
Transmit "txstr" from COM<portnumA> as unicode. Hence twice number of bytes sent. Proceed immediately to next action

WAITDATA <timeoutMs> <portnum> <channel> <string>
Will wait for a max period of time to see if the receive buffer in 'portnum' contains the “string” data after extraction from appropriate packet. Useful for multipoint protocol testing when received data will get fragmented

Note:The timeout value is scaled according to the percent value specified by the most recent TIMESCALAR statement

WAITCHANNEL <timeoutMs> <portnum> <channel> <string>
Will wait for a max period of time to see if data has been recevined in a channel as defined for the PORT_OC logicl interface. The receive buffer of the channel in 'portnum' contains the “string” data after extraction from appropriate packet. Useful for text based packet protocol testing when received data will get fragmented

Note:The timeout value is scaled according to the percent value specified by the most recent TIMESCALAR statement

WAITMULTI <timeoutMs> <portnum> [str0|str1|str2…]
Will wait for a max period of time to see if the receive buffer in 'portnum' contains one of the substring "strN". If a match is found then see statement MULTIINDEX to get the 0 based index number of the matching string.
Note: This command only works if portnum was opened using PORT_MPOINT

Note: The timeout value is scaled according to the percent value specified by the most recent TIMESCALAR statement

WAITPACKET <timeoutMs> <portnum> <varname>
Will wait for a max period of time to see if the receive buffer in 'portnum' contains at least one packet and if so will extract it from the response buffer and save it in the variable varname1.
If packet is written to variable then zero flag is set so that IFNZ/JUMPNZ will succeed.

Note:The timeout value is scaled according to the percent value specified by the most recent TIMESCALAR statement

WAITRESP <timeoutMs> <portnum> <respstr>
Will wait for a max period of time to see if the receive buffer in 'portnum' contains the substring "respstr"

Note:The timeout value is scaled according to the percent value specified by the most recent TIMESCALAR statement

WAITRESPEX <timeoutMs> <portnum> <respstr> <optional_varname>

Will wait for a max period of time to see if the receive buffer in 'portnum' contains the substring "respstr".
If the substring is found and the <optional_varname> parameter is absent, then all data up to and including that substring is deleted from the low level buffer.
If however, the <optional_varname> parameter has been specified, then in addition to deleting the data up to the match string, a copy is saved in that variable.

Note:The timeout value is scaled according to the percent value specified by the most recent TIMESCALAR statement

WAITTIEVENT <timeoutMs> <portnum> <varname> <matchstr>

Will wait for a max period of time to see if the receive buffer in 'portnum' contains at least one packet and if so will extract it from the response buffer and save it in the variable varname1 IF and ONLY IF the packet contains the "matchstr" specified. If packet is written to variable then zero flag is set so that IFNZ/JUMPNZ will succeed.

Note:The timeout value is scaled according to the percent value specified by the most recent TIMESCALAR statement

WAITTIMEOUT <timeoutMs> <portnum> <respstr> <optional_varname>

Reverse logic of WAITRESPEX in the sense that there is a failure if the <respstr> is received within the <timeoutMs> period specified.
Will wait for a max period of time to ensure that the receive buffer in 'portnum' does not contain the substring "respstr".
If the <optional_varname> parameter has been specified, then in addition to deleting all the data in the rx buffer is saved in that variable.

WAITUNTIL <timeoutMs> <portnum> <respstr> <optional_varname>

This statement will NOT generate a PASS or FAIL outcome, but will result in the Z flag being updated.
If the string match occurs then the Z flag is true. If the timeout occurs then the Z flag is false.

If the <optional_varname> parameter has been specified, then in addition to deleting all the data in the rx buffer is saved in that variable.

APPLICATION EXIT CODE (WHEN QUIT IS SPECIFIED IN COMMAND LINE)

0	PASS
>0	FAIL and the value specifies the number of failures
-1	A *smart*BASIC application failed to compile
-2	Comport could not be opened
-3	PORT statement failed with *Host Creation Error*
-4	PORT statement failed with *Serial Port X failed to open*
-5	PORT statement failed with *Comms Parameter X is invalid*
-6	PORT statement failed with *X is not a valid name for a serial port*
-7	PORT statement failed with *Logical Port X cannot be specified*
-8	PORTCLOSE statement failed with *Logical Port Host Type unexpected*
-9	PORTCLOSE statement failed with *Logical Port X invalid*

-13	PORT statement failed with *Host Creation Error*
-14	PORT statement failed with *Serial Port X failed to open*
-15	PORT statement failed with *Comms Parameter X is invalid*
-16	PORT statement failed with *X is not a valid name for a serial port*
-17	PORT statement failed with *Logical Port X cannot be specified*
-23	BTCONSOLE statement failed with *BtConcole Host Creation Error*
-24	BTCONSOLE statement failed with *BtConsole failed to open*
-25	BTCONSOLE statement failed with *Logical Port X cannot be specified*
-33	PIPE statement failed with *Host Creation Error*
-34	PIPE statement failed with *Pipe failed to open*
-37	PIPE statement failed with *Logical Port X cannot be specified*
-43	PORT_OC statement failed with *Host Creation Error*
-44	PORT_OC statement failed with *Serial Port X failed to open*
-45	PORT_OC statement failed with *Comms Parameter X is invalid*
-46	PORT_OC statement failed with *X is not a valid name for a serial port*
-47	PORT_OC statement failed with *Logical Port X cannot be specified*
-1000	Generic error