EFILive BBX Scripting Reference

Paul Blackmore

EFILive BBX Scripting Reference

First published
9 June 2022

Revised
17 June 2022

Contents

Prerequisites

Intended Audience

EFILive tuners who need to customize and/or enhance FlashScan and/or AutoCal devices with additional functionality, not supplied by EFILive.

Computer Knowledge

It is expected that readers have a basic understanding of:

- The Windows operating system;
- Starting and using Windows applications.

Lua Scripting

To make use of these scripting capabilities you must have a good understanding of the Lua programming language.

https://www.lua.org/

https://www.lua.org/manual/5.4/

Supported Hardware

BBX scripting is only available on V3 devices.

FlashScan V1, FlashScan V2 and AutoCal V2 do not support BBX scripting.

Introduction

EFILive BBX Scripting Reference

Standalone Operation

BBX scripting allows you to add custom functions to the operation of FlashScan V3 and AutoCal V3 devices.

Used in conjunction with a custom menu, you can customize the look, feel and operation of the V3 device.

Example Usage

You could add an option to display and/or modify tire size.

You could add an option to display and/or modify the VIN.

Limitations

Memory

The Lua environment is limited to 256KB of memory. If that memory limit is exceeded then the Lua script will fail. You can monitor the memory usage using the bb.MemoryStats() function.

Modules

Only the following Lua modules are available:

- base
- string
- math
- table
- io
- utf8

The following Lua modules are not available:

- coroutine
- debug
- os
- package

BBX Scripting

Installation

Version Compatibility

Software version must be EFILive V8.3.25 or later.

Firmware version of FlashScan/AutoCal V3 devices must be V3.0.93 or later.

Usage

See the sample files:

\Documents\EFILive\V8\User Defined Menus\SampleCustomMenu.txt

\Documents\EFILive\V8\User Defined Menus\ Lua\CustomFunctions.Lua

\Documents\EFILive\V8\User Defined Menus\Lua\HelperFunctions.Lua

Lua Compiler

Download the Lua compiler here:

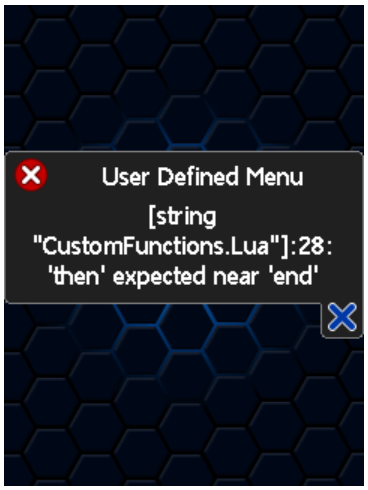
http://luabinaries.sourceforge.net/download.html

Specifically, EFILive is currently using Lua 5.4 so use this direct download link:

https://sourceforge.net/projects/luabinaries/files/5.4.2/Tools Executables/lua-5.4.2_Win32_bin.zip/download

Lua Syntax Errors

When a custom menu and associated Lua source files are installed onto a FlashScan or AutoCal device, the Lua source files are compiled by the Lua instance in the firmware. If a syntax error is detected in any of the Lua source files a message like the one below will be displayed

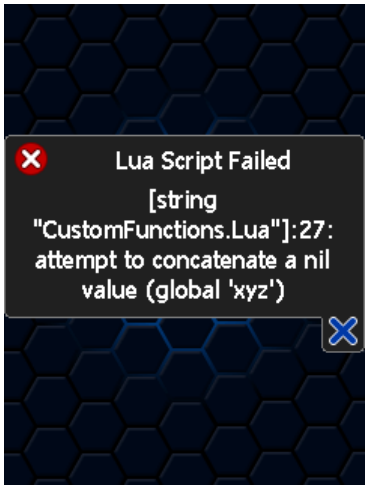


That message means the error is in the file CustomFunctions.Lua at line 28.

It is recommended that you pre-compile any Lua source files using the Luac54.exe compiler and only distribute the compiled binaries. A custom menu will be loaded faster if the Lua files are already compiled.

Lua Runtime Errors

When a custom Lua function is executed, any errors will be displayed like this:



That message means the error occurred in the file CustomFunctions.Lua at line 27.

Variable Scope

It is considered good practice to limit the scope of Lua variables. You can do that by declaring and initializing variables as “local”, like this:

```
function Memory()
  --- Example function to show how to display the device's Lua memory usage.
  --- 256KB of memory is reserved for Lua.
  --- If Lua exceeds 256KB of memory use, scripts will fail.
  local mt,mf,mu,mm = bb.MemoryStats()

  local msg = string.format('Total: %d\nFree: %d\nUsed: %d\nMaximum: %d',mt,mf,mu,mm)
  bb.Dialog(bb.Style_Info,0,'Memory Usage',msg,bb.Button_Cross,bb.Wait_Forever)
end
```

By declaring variables as “local”, the variables only exist while that function is running. Any memory used by those variables is automatically returned to Lua as soon as the variable goes out of scope (i.e., when the function returns to the caller). Lua can then re-use that memory for other variables.

If you don’t declare a variable as “local” then it is a global variable and it will persist in the Lua environment until FlashScan/AutoCal is rebooted.

Global variables (that are not declared as local), can be useful for sharing data between Lua functions that are invoked by different menu items.

You can manually discard a global variable and free up the memory it uses by setting its value to nil.

Built-in Variables

Device Info

Name	Value
bb.Serial	Device's serial number.
bb.License	Device's license number.
bb.Version[1]	Device's major version.
bb.Version[2]	Device's minor version.
bb.Version[2]	Device's release number.
bb.Version[2]	Device's build number.

Return Values

Name	Value	Description
bb.None	0	Value: 0, Use for options such as No protocol, No Baudrate, No Icon, No Input restrictions, etc.
bb.OK	0	Value: 0, use for checking bb.() functions' results.

OBDII Protocol

Name	Value	Description
bb.VPW	1	GM's VPW (Variable Pulse Width) protocol.
bb.CAN	2	CAN (Controller Area Network) protocol

CAN Protocol

Name	Value	Description
bb.CAN_J1979	1	SAE-J1979 CAN protocol used in light and medium duty vehicles.
bb. CAN_J1939	2	SAE-J1939 CAN protocol used in heavy duty vehicles.

CAN Baud

Name	Value	Description
bb.CAN_250K	250000	CAN baud rate at 250Kbps.
bb.CAN_500K	500000	CAN baud rate at 500Kbps.
bb.CAN_1000K	1000000	CAN baud rate at 1Mbps.

Dialog Styles

Name	Value	Description
bb.Style_None	0	No icon is displayed.
bb.Style_Error	1	Error icon is displayed.
bb.Style_Info	2	Info icon is displayed.
bb.Style_Ask	3	Ask icon is displayed.
bb.Style_Warn	4	Warn icon is displayed.
bb.Style_Dtc	5	DTC icon is displayed.
bb.Style_Folder	6	Folder icon is displayed.

Dialog Timeout

Name	Value	Description
bb.Wait_Forever	0xFFFFFFFF	Dialog will not timeout.

Input Styles

Name	Value	Description
bb.Input_None	0	All characters are allowed
bb.Input_Alpha	1	Only 'A'..'Z' and 'a'..'z' are allowed.
bb.Input_Uint	2	Only '0'..'9' are allowed.
bb.Input_Int	3	Only '-' and '0'..'9' are allowed.
bb.Input_Float	4	Only '-', '.' and '0'..'9' are allowed.
bb.Input_AlphaNum	5	Only '0'..'9', 'A'..'Z' and 'a'..'z' are allowed.
bb.Input_DTC	6	Diagnostic trouble code: Xhhhh where X is one of 'P','B','C','U' and h is a hexadecimal digit.

Where numeric values allow non digits to be entered, such as a leading minus sign or a decimal point, no restriction is enforced on where or how many of those non-digit symbols the user can enter.

For example, for input style 4 (for floating point values) the user could enter something like this: **12.34.56-** and the input dialog would happily accept it, even though it is not a valid floating-point value.

So, you must validate the user's numeric input to ensure it is a real number prior to using the numeric result.

Dialog Buttons

The bb.Button_xxx values may be added together to cause multiple icons to appear at the bottom of the dialog.

Name	Value	Hex	Description
bb.Button_Cross	2	0x00000002	Cross
bb.Button_Left	4	0x00000004	Left
bb.Button_Back	8	0x00000008	Back
bb.Button_Up	16	0x00000010	Up
bb.Button_Down	32	0x00000020	Down
bb.Button_Inc	64	0x00000040	Inc
bb.Button_Dec	128	0x00000080	Dec
bb.Button_Right	256	0x00000100	Right
bb.Button_Tick	512	0x00000200	Tick
bb.Button_CtrlNo	131072	0x00020000	No icon displayed
bb.Button_CtrlLeft	262144	0x00040000	No icon displayed
bb.Button_CtrlRight	16777216	0x01000000	No icon displayed
bb.Button_CtrlYes	33554432	0x02000000	No icon displayed

The set of buttons assigned to a dialog determines which buttons will cause the dialog to return to the caller. If the user presses a button that is not in the dialog's button set, then the button press is ignored and a short beep is sounded to warn the user that the button is not valid.

Dialog Results

Only results that are included in the set of dialog buttons can be returned from a dialog.

Name	Value	Description
reserved	1	
bb.Result_Tick	2	✔ was pressed
bb.Result_Up	4	▲ was pressed
bb.Result_Cross	3	✖ was pressed
bb.Result_Down	5	▼ was pressed
bb.Result_Back	6	◀ was pressed
bb.Result_Left	7	◀ was pressed
bb.Result_Right	8	▶ was pressed
reserved	9	
bb.Result_CtrlLeft	10	Fn ◀ was pressed
bb.Result_CtrlRight	11	Fn ▶ was pressed
bb.Result_CtrlYes	12	Fn ✔ was pressed
bb.Result_CtrlNo	13	Fn ✖ was pressed
bb.Result_Timeout	14	The dialog timed out before the user pressed a button.

Special characters

Escape sequences

Embed the following escape sequences in text to display the associated special characters.

Char	Esc		Char	Esc		Char	Esc
◀	\x10		λ	\x1B		µ	\B5
▶	\x11		™	\x99		¼	\BC
▲	\x12		©	\xA9		½	\BD
▼	\x13		®	\AE		¾	\BE
✖	\x14		°	\B0			
✔	\x15		²	\B2			
Fn	\x16		³	\B3			

Built-in Functions

EFILive has provided a number of functions that may be called from Lua scripts. These functions allow the scripts to interact with the device, the SD Card and the connected vehicle.

Dialogs

bb.Dialog(style,options,caption,msg,buttons,timeout)		
Displays a message in a dialog and provides one or more buttons for the user to select.		
Parameters	style	The dialog icon/style type: See bb.Style_xxx
	options	Reserved, set to 0.
	caption	The text to display at the top of the dialog.
	msg	The text to display in the body of the dialog Text wraps automatically. Use \n to force a new line.
	buttons	One or more buttons to be displayed as user selectable options at the bottom of the dialog: See bb.Button_xxx
	timeout	Timeout in milliseconds. If the user does not select an option within the timeout period, the dialog box returns the value bb.Result_Timeout (14). Use bb.Wait_Forever for no timeout.
Returns(1)	result	Button that user selected: See bb.Result_xxx

bb.Input(style,options,caption,msg,value,length,timeout)		
Displays a message in a dialog and a single value that the user can edit using the arrow keys.		
Parameters	style	The dialog icon/style type: See bb.Input_xxx
	options	Reserved, set to 0.
	caption	The text to display at the top of the dialog.
	msg	The text to display in the body of the dialog Text wraps automatically. Use \n to force a new line.
	value	The initial value to be edited by the user. The value to be edited is placed on the line after the last line of the msg.
	length	The number of characters/digits that the value requires.
	timeout	Timeout in milliseconds. If the user does not select an option within the timeout period, the dialog box returns the value bb.Result_Timeout. Use bb.Wait_Forever for no timeout.
Returns(2)	result	Button that user selected: One of bb.Result_Tick bb.Result_Cross
	value	The value entered by the user.

bb.Select(options,caption,itemList,itemCount,index,timeout)		
Displays a list of options in a dialog from which the user can select using the up/down arrow keys.		
Parameters	options	Reserved, set to 0.
	caption	The text to display at the top of the dialog.
	itemList	List of items to be selected. Each item is on a new line. Use \n to force a new line.
	itemCount	Number of items in the list.
	index	Selected item when the dialog opens.
	timeout	Timeout in milliseconds. If the user does not select an option within the timeout period, the dialog box returns the value 14. Use bb.Wait_Forever for no timeout.
Returns(2)	result	Button that user selected: One of bb.Result_Tick bb.Result_Cross
	value	The index selected by the user.

Time

bb.Hourglass()	
Displays the cloud icon to indicate that the user should wait for a short amount of time. You should consider displaying the cloud icon immediately after a user presses a button that takes longer than 250ms seconds to respond visually. It is disconcerting for the user to press a button and not see an immediate response. The user is likely to press the button again.	
Parameters	none
Returns	none

bb.Wait(timeout)		
Pause script.		
Parameters	timeout	The script will pause for this many milliseconds.
Returns	none	

Debugging

bb.MemoryStats()		
Retrieves the current memory usage statistics in bytes.		
Parameters	None	
Returns(4)	Total	Total memory available to Lua.
	free	Currently free memory.
	Used	Currently used memory.
	max	Highest amount of memory used so far.

bb.Trace(show)		
Save trace file containing up to the last 100 OBDII messages sent/recv by the script. Only up to the first 8 bytes of each message are stored.		
Parameters	Show	true: Show filename to user. false: Do not show filename.
Returns(2)	result	0: Success. Non-zero: Error code.
	filename	Name of the file that was saved.

Send and Receive

bb.Send(id,data,timeout)		
Send an OBDII message using the current protocol.		
Parameters	id	CAN ID of the destination node. Or VPW ID of the destination node.
	data	String containing the bytes to be sent.
	timeout	Timeout in milliseconds.
Returns(1)	result	0: Success. Non-zero: Error code.

bb.Recv(timeout)		
Receives an OBDII message using the current protocol.		
Parameters	timeout	Timeout in milliseconds.
Returns(3)	result	0: Success. Non-zero: Error code.
	id	CAN ID of the msg. Or VPW ID of the msg.
	msg	String containing the bytes received.

OBDII Protocol

bb.AutoDetectProtocol()		
Auto detects the current protocol.		
Parameters	none	
Returns(4)	result	true: protocol detected. false: protocol not detected.
	protocol	Protocol detected. • bb.None • bb.VPW • bb.CAN
	canProtocol	CAN protocol detected • bb.None • bb.CAN_J1979 (light duty) • bb.CAN_J1939 (heavy duty)
	canBaud	CAN baud rate detected • bb.None • bb.CAN_250K • bb.CAN_500K • bb.CAN_1000K

bb.GetProtocol()		
Gets the current protocol.		
Parameters	none	
Returns(1)	protocol	Current protocol.

bb.SetProtocol(protocol)		
Sets the current protocol.		
Parameters	protocol	Protocol to set
Returns(1)	protocol	Current protocol.

bb.GetCanProtocol()		
Gets the current CAN protocol.		
Parameters	None	
Returns(1)	canProtocol	Current CAN protocol.

bb.SetCanProtocol(canProtocol)		
Sets the current CAN protocol.		
Parameters	canProtocol	CAN protocol to set
Returns(1)	canProtocol	Current CAN protocol.

bb.GetCanBaud()		
Gets the current CAN baud rate.		
Parameters	None	
Returns(1)	canBaud	Current CAN baud rate.

bb.SetCanBaud(canBaud)		
Sets the current CAN baud rate.		
Parameters	canBaud	CAN baud rate to set
Returns(1)	canBaud	Current CAN baud rate.