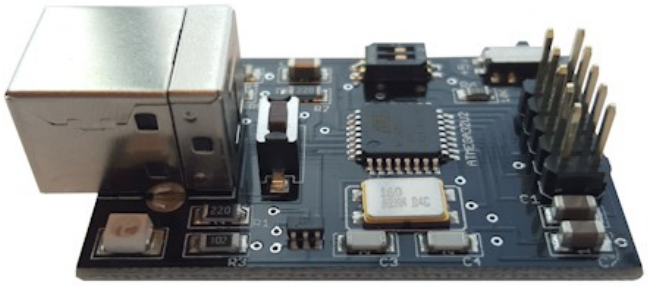
Embedded Computers

For professionals



FlashcatUSB

SCRIPT ENGINE DOCUMENTATION

Website: www.embeddedcomputers.net/products/FlashcatUSB/
Support email: contact@embeddedcomputers.net
Last updated: February, 2019

FLASHCAT

QUICK LOOKUP

GETTING STARTED

Before you begin, please make sure you are using the newest software/firmware for your device. You can download the newest software from EmbeddedComputer's website.

The software includes many scripts. Advice for beginners, just take a look at some of those to get an idea of how to write/use them.

A script file is just a plain text document. You can open and edit one using Notepad. The contents of a script file are made up of commands, labels, and sub procedures (called events). When a script file is executed, any line that is not in an event block will be executed. The purpose of a script is to accomplish more complex operations that may not otherwise be performed using the standard GUI. Script files are also ideal for production environments.

Basic Syntax

DataTypes (Integer, String, Data)

Variables

Events / Functions

Conditional Statements (IF … ELSE … ENDIF)

Loops / Iterations (FOR … ENDFOR)

AUTORUN Feature

Basic Syntax

FlashcatUSB scripts use a very familiar console/script syntax. Each line contains a statement that can be a command to execute, a condition (IF etc.), a flow control statement, or the start of an Event (a function or sub procedure). To put comments into your script file (statements which have no affect), use the # symbol. Any characters proceeding will not be evaluated by the script engine.

The basic flow-control statements are: LABEL, GOTO, EXIT, RETURN.

The GOTO statement can be used to control which line to execute. Generally, the software starts executing at the top of the file and proceeds down. To change the location, you can use the GOTO statement followed by a location that is indicated by using a label. A LABEL is a user-specified word that ends with a colon. For example:

```
Var1 = 10
GOTO SKIP_MINVAL
Var1 = Var1 - 5
SKIP_MINVAL:
msgbox(Var1)        #Var1 = 10
```

So as the script executes, when it reaches the GOTO keyword, the engine will then search the document (forwards and backwards), for a label that contains the name “SKIP_MINVAL”. Once found, execution will begin there. Because the script engine will search both ways, you can also use GOTO keywords to create loops.

The EXIT keyword can be used to leave an event or function, or to exit out of a condition segment (IF or FOR block for example). The syntax usage is EXIT, EXIT EVENT, or EXIT SCRIPT. If you run exit script, no matter what event you are in, the entire script will stop executing. For example:

```
If (VarInt = 10) #This does a compare to see if VarInt is 10
   SomeFunction()
   Var2 = 0x40
   exit
   Var3 = 0x40     #This will not be executed
Endif
```

When the script reaches the exit command, it will then exit out of this IF statement and proceed to execute the next line after the EndIf statement.

Data Types

There are 4 main data types that you can work with. These are:

Bool - is a value of either TRUE or FALSE

String - is a value of ASCII readable characters, specified by using quotation marks.

Integer - is a 32 bit (unsigned) number. Hex values are automatically converted.

Data - is an array of bytes. Can also be specified with 0x80;0x81;0x82; etc.

Arithmetic Operators

The following operators are supported:

+ - Addition. Adds two integers, combines two strings, or combines two data arrays.

- - Subtract. Only compatible with Integer data types.

/ - Divide. Only compatible with Integer data types.

***** - Multiple. Only compatible with Integer data types.

<< - Shift left. Shits an integer to the left by a number of bits.

>> - Shift right. Shifts an integer to the right by a number of bits.

Logical Operators

== - compares two operands and returns BOOL TRUE if they are exact.

& - logical AND

| - logical OR

Variables

A variable is a name that you assign an object. You can assign a string, data, integers, or boolean values. For example:

```
ThisVar = "Hello World"
```

Will now create a variable named ThisVar whose string value is "Hello World". To create a data array use ";" after each byte:

```
MyData = 0x01;0x02;0x03;0x04;0x05;0x06;
```

If you assign a variable 4 or less bytes, the variable will auto convert to a Integer type instead of a Data type. To create a boolean variable:

```
DoVar = True
```

And to create an integer:

```
VarInt = 470
```

Integer variables are able to be added or subtracted. String and Data variables can be combined.

```
VarInt = 5
VarInt += 10
msgbox(VarInt)        #this will produce the result of 15
```

For strings and data, use the operand "&", for example:

```
VarStr = "Hello "
```

```
VarStr = VarStr + "World!"
msgbox(VarStr)        #Will produce "Hello World!"
MyData = 0x01;0x02;0x03;0x04;0x05;
MyData = MyData + 0x06;0x07;
msgbox(hex(MyData))     #Will produce "0x01020304050607"
```

The hex command converts the data array into a hex string that can be printed.

DATA arrays can also have index arguments specified to read data from the array and use it like an Integer that contains a byte.

```
VAR1 = 0x01;0x02;0x03;
VAR2 = VAR1(1)
msgbox(VAR2)            #Will print “2”
```

To write a byte to a DATA array, you can also do this:

```
VAR1 = 0x01;0x02;0x03;
VAR1(0) = 0xFF
msgbox(String.Hex(VAR1(0)))              #Outputs “0xFF”
```

Events / Functions

An Event is similar to a function that you may be familiar with. You can treat it like a “function”, you can create events and then call them like functions, and even return a value. Events can also be assigned to GUI elements, such as buttons. So when you click a button you created, the engine will run the commands that are in the Event that you assigned to that button.

Events are very useful. You can pass variables to events and retrieve values from events. When you pass a variable or value to an event, the event will create a new variables for each argument passed. These new variables will be named $1, $2, $3 and so on for each variable passed.

Within an Event block, you can exit the block and create a new variable using the **RETURN** keyword.

To create an event or function, use the CreateEvent keyword followed by a parameter specifying the name of the event/function. And to specify the end of the Event, use the EndEvent keyword.

```
EchoToMsgBox("Hello World")
CreateEvent(EchoToMsgBox)
    msgbox($1)
```

```
EndEvent
```

This code sample popup a message box saying "Hello World" when executed. You can also use events like functions to parse information an use the event like you would a command function. For example:

```
msgbox(CombineString("Hello"," World"))
CreateEvent(CombineString)
    StrVar = $1 + $2
    Return StrVar
EndEvent
```

The output from this will produce a message box that says “Hello World”.

Conditional Statements

To execute code based on a condition, you can use the IF keyword followed by a statement that can be evaluated to be either true or false. The syntax is IF (STATEMENT). Optionally, you can prefix the statement with the NOT keyword to indicate that the statement should evaluate to the opposite. This is similar to the “IF, ELSE, ENDIF” of other programming languages.

```
If (5 > 2)
    msgbox("This will be executed")
Else
   msgbox("This will not")
EndIf
```

The condition statement (5 > 2) is evaluate and found to be true. You can also use Events that return TRUE or FALSE. If you precede the condition statement with the "not" keyword, what ever the statement is evaluated at, the opposite will happen. You can also use the "!" character for the same effect.

```
If not (GetValue() > 10)
    msgbox("This will be executed")
EndIf
CreateEvent(GetValue)
    retVar = 5
    return retVar
EndEvent
```

In the above example, you can create a function named GetValue, by specifying it using the CreateEvent keyword. Inside the event block, you can then run commands or other syntax

and then use the Return keyword to return a value to the calling line, in this case, the IF statement that compares the return value to be greater than 10.

You can also use an **IF CONDITION** with the keyword **NOTHING** to check to see if a variable exists. For example

```
If not (ObjectVar==Nothing)
        msgbox("variable exists!")
Else
        msgbox("Variable does not exist!")
EndIf
```

Loops / Iterations

To do repetitive tasks, you can use a FOR loop. This is specified by using the FOR keyword followed by parameters specifying a variable name, starting value, and ending value. Optionally, you can specify a step value (an integer to increase the counter by).

```
For (i = 0 to 9)
        msgbox("We are on loop number: " & i)
EndFor
```

This will iterate 50 times (0, 2, 4, 6, ...)

```
For (i = 0 to 98) Step 2
        msgbox("We are on loop number: " & i)
EndFor
```

AUTORUN Feature

If you want the software to automatically load a script when connected to a JTAG board or SPI device, you can use this feature. Since some devices share the same CPU ID code, and you may want to have different device scripts, you can use the autorun feature. To do so, edit or create the Autorun.ini file located in the Scripts folder. Each line (not commented out) represents one device script. The format is:

<JTAG_ID or JEDEC_ID>:<SCRIPT_NAME>:<DEVICE_NAME>

Add as many scripts as you need and when you run the software, when it connects to a device it will load all of the scripts that match the CPU ID or JEDEC ID. To change scripts, simply change the menu item on the drop down list. For your convenience the last script executed will be remembered for future use.

LIST OF CONSOLE OR SCRIPT COMMANDS

The following is a complete list of commands that are built into the FlashcatUSB script engine. You can execute these either in a script file or from the software's console window. Some commands will output information to the console, others will not. Also note that for the memory commands, if you have initiated more than one memory device, you can access each device by using parameters with an index, for example, memory(0).read will perform the read operation from the first memory device; memory(1).read will do the same from the second device, and so on.

I/O Commands (functions for reading/writing to your hard drive)
Memory Commands (functions for reading/writing to your SPI/CFI memory)
GUI Commands (for creating tabs, buttons, and other GUI elements)
JTAG Specific Commands (commands to be used only in JTAG mode)
SPI Specific Commands (commands to be used only in SPI mode)
Miscellaneous Commands (All other commands/functions supported)

STRING Commands

Command:	upper
Parameters:	String
Returns:	String
Description:	Inputs a string and converts it all uppercase.
Examples:	String.upper(“hello”) #ouputs "HELLO”

Command:	lower
Parameters:	String
Returns:	String
Description:	Inputs a string and converts it to all lower case.
Examples:	String.lower(“ANIME”) #ouputs "anime"

Command:	hex
Parameters:	Integer
Returns:	String
Description:	Converts a integer value into a hex string
Examples:	String.hex(255) #ouputs "0xFF"

Command:	length
Parameters:	String
Returns:	Integer
Description:	Returns the number of bytes the string requires in memory.
Examples:	String.length(“zenzenzense”) #returns 11

Command:	ToInt
Parameters:	String
Returns:	Integer
Description:	Converts a string that contains a numeric value to an unsigned integer.
Examples:	var_int = String.ToInt(“1024”)

Command:	FromInt
Parameters:	Integer
Returns:	String
Description:	Converts a an integer value or variable to a string.
Examples:	var_str = String.FromInt(1024)

DATA Commands

Command:	New
Parameters:	Integer, Data (optional)
Returns:	Data
Description:	Creates a new data array with a specific number of bytes. The parameter specifies the size of the array. The second parameter will set the initial data to create

Examples:	DVAR = Data.New(128,0xFF) #Creates an array with 128 bytes
	DVAR = Data.New(512,0xAA;55) #Alternating bytes

Command:	Compare
Parameters:	Data, Data
Returns:	Bool
Description:	Compares two data arrays and returns true if they both exist and are the same.
Examples:	Result = Data.Compare(VAR1,VAR2)

Command:	Length
Parameters:	Data
Returns:	Integer
Description:	Returns the size of the data array.
Examples:	size = Data.Length(VAR1)

Command:	Resize
Parameters:	Data, Integer, Integer (optional)

Description:	Resizes a data array. First parameter is the data array to resize, the second is the offset within the array, and the last (optional) argument is the number of bytes to copy.
Examples:	Data.Resize(VAR1)

Command:	Word
Parameters:	Data, Integer
Returns:	Integer
Description:	Copies a word (4 bytes) of data from a DATA array at a specific offset and saves it as an Integer.
Examples:	Var1 = Data.Word(data_array,0)

Command:	Hword
Parameters:	Data, Integer
Returns:	Integer
Description:	Copies a half-word (2 bytes) of data from a Data array at a specific offset and saves it as an Integer.
Examples:	Var1 = Data.Hword(data_array,0)

Command:	ToStr
Parameters:	Data
Returns:	String
Description:	Converts the data array to hex string.
Examples:	Var1 = Data.ToStr(data_array)

Command:	Copy
Parameters:	Data, Integer, Integer (optional)
Returns:	Data
Description:	Copies a section of a Data array and creates a new array. The first parameter is the Data variable, the second is the offset within the array, and the third is the length of data to copy, otherwise it will copy until the end.
Examples:	Var1 = Data.Copy(data_array,0,128) #Copies 128 bytes to a new array

I/O commands

Command:	Open
Parameters:	String (optional), String (optional), String (optional)
Returns:	Data
Description:	Prompts the user for a file and then reads the file from disk and returns a data

	variable. First optional parameter is the title of the window and the optional second is the standard file filter to use. The third optional argument can specify a sub-directory to start in.
Examples:	MyData = IO.Open("Choose file", "Firmware files (*.bin)|*.bin")

Command:	Save
Parameters:	Data, String (optional), String (optional)
Syntax:	Data variable to write, title prompt, default save name
Description:	Prompts the user to save a data variable to the hard drive.
Examples:	IO.Save(MyData,"Where to save?","fileread.bin")

Command:	Read
Parameters:	String
Returns:	Data
Description:	Reads a file from the hard drive. The first parameter is the name of the file (in relation to where FlashcatUSB.exe is located).
Examples:	MyData = IO.Read(“Scripts\EEPROM.bin”)

Command:	Write
Parameters:	Data, String
Description:	Writes data to the hard drive. The first parameter is a data array variable, the second is the location where you want to save the file. This command does not prompt the user.
Examples:	IO.Write(MyData,“Scripts\EEPROM.bin”)

Memory commands

Command:	Memory.Name
Parameters:	None
Returns:	Integer
Description:	Returns the name of the memory device.
Examples:	flash_name = Memory(0).Name

Command:	Memory.Size
Parameters:	None
Description:	Returns the the number of bytes in the given memory device.
Examples:	flash_size = Memory(0).Size

Command:	Memory.Write
Parameters:	Data, Integer, Optional Integer
Syntax:	Data object to write, flash address offset, optional length
Description:	Writes a data variable to the flash device. Works for both CFI and SPI flash devices, but please note you must have already initiated the flash.
Examples:	Memory.Write(dataVar,0,256) #Writes

Command:	Memory.Read
Parameters:	Integer, Integer, Optional Bool
Returns:	Data
Description:	Reads data from the flash device. First parameter is the flash offset, the second is number of bytes to read. The third option disables status updates.
Examples:	dataVar = Memory.Read(0,512) #Reads 512 bytes

Command:	Memory.ReadString
Parameters:	Integer
Returns:	String
Description:	Reads a string from the offset specified on the flash device. Returns nothing if error or string not found.
Examples:	dataStr = Memory.ReadString(0x5000)

Command:	Memory.ReadVerify
Parameters:	Integer, Integer
Returns:	Data
Description:	Similar to ReadFlash(), this function actually does it twice and compares the result, and if needed verifies all data to ensure that the data read is 100% accurate. Returns nothing if verification failed. This function is preferred over ReadFlash where the integrity of the data is vital.
Examples:	dataVar = Memory.ReadVerify(0,512) #Reads 512 bytes

Command:	Memory.SectorCount
Returns:	None
Description:	Returns the number of sectors (some times called blocks) of a device.
Examples:	sectors = Memory.SectorCount()

Command:	Memory.SectorSize
Returns:	Integer
Description:	Returns the number of bytes of a given sector. Some devices have different

	sector sizes, while others have uniform sizes.
Examples:	sector_size = Memory.SectorSize(0)

Command:	Memory.EraseSector
Parameters:	Integer
Returns:	None
Description:	Erases the specified flash sector.
Examples:	Memory.EraseSector(0) #Erases the first sector

Command:	Memory.EraseBulk
Parameters:	None
Returns:	Nothing
Description:	Erases the entire flash memory
Examples:	Memory.EraseBulk()

Command:	Memory.Exist
Parameters:	None
Returns:	Bool
Description:	Returns true if a memory device at a given index has been created.
Examples:	Memory(2).Exist()

GUI commands

Command:	Tab.Create
Parameters:	String
Returns:	Integer
Description:	Creates a application specific tab. Returns the index of the tab.
Examples:	Tab.Create("My Device")

Command:	Tab.AddGroup
Parameters:	String, Integer, Integer, Integer, Integer
Syntax:	Name of group, (X-axis), (Y-axis), Length, Height
Description:	Creates a group box on the tab.
Examples:	Tab.AddGroup("Feature",10,10,420,140)

Command:	Tab.AddBox
Parameters:	String, String, Integer, Integer

Description:	Creates a input box on your tab.
Examples:	Tab.AddBox("BXNAME","default text",30,110)

Command:	Tab.AddText
Parameters:	String, String, Integer, Integer
Description:	Creates a text label on your tab.
Examples:	Tab.AddBox("txtName","What to say",30,110)

Command:	Tab.AddImage
Parameters:	String, String, Integer, Integer
Description:	Adds a image to your tab from the specified file (in your scripts folder)
Examples:	Tab.AddImage("ImgName","logo.gif",20,20)

Command:	Tab.AddButton
Parameters:	Event, String, Integer, Integer
Description:	Adds a button to your tab. The specified event is called when the user clicks on the button.
Examples:	Tab.AddButton(HelloWorld,"Click Me!",20,20)

Command:	Tab.AddProgress
Parameters:	Integer, Integer, Integer
Description:	Adds a progress bar to your form. This bar will then be automatically updated via internal functions that you call (selected ones that might take time to process). The parameters are x-axis, y-axis, and bar width.
Examples:	Tab.AddProgress(20,92,404)

Command:	Tab.Remove
Parameters:	String
Description:	Removes any previously added object from your tab.
Examples:	Tab.Remove("ImgName")

Command:	Tab.SetText
Parameters:	String, String
Description:	Changes the text of any previously created object
Examples:	Tab.SetText("txtName","hello world")

Command:	Tab.ButtonDisable
Parameters:	String (Optional)

Description:	Disables a button so the user can not click it and run the event. The input is the name of the button, but if omitted, then all user created buttons will be affected.
Examples:	Tab.ButtonDisable("btName")

Command:	Tab.ButtonEnable
Parameters:	String (Optional)
Description:	Enables the button (if you had it disabled). The input is the name of the button, but if omitted, then all user created buttons will be affected.
Examples:	Tab.ButtonEnable("btName")

SPI commands

Command:	SPI.Clock
Parameters:	Integer
Description:	Used to set the hardware SPI clock (in MHZ). For Classic, compatible speeds are 8, 4, 2, and 1. For Professional they are 5, 8, 10, 12, 15, 20, 24, and 30.
Examples:	SPI.Clock(8)

Command:	SPI.Order
Parameters:	String
Description:	Used to set the bit order for all SPI commands. For most significant bit, use "MSB" for least significant bit use "LSB".
Examples:	SPI.Order("MSB")

Command:	SPI.Mode
Parameters:	Integer
Description:	Used to set the SPI device mode. Supported modes 0, 1, 2, 3.
Examples:	SPI.Mode(0)

Command:	SPI.Database
Parameters:	Bool
Description:	Prints the entire list of supported SPI devices and size in bits. Optional parameter to also display the JEDEC ID.
Examples:	SPI.Database(true)

Command:	SPI.GetSR
Parameters:	Integer (optional)
Returns:	Data

Description:	Prints and returns the value of the status register. Optional parameter can set the number of bytes to read (default is 1).
Examples:	SPI.GetSR(1)

Command:	SPI.SetSR
Parameters:	Data
Description:	Writes data to the status register. You can write a single byte, or multiple bytes.
Examples:	SPI.SetSR(0xF0)

Command:	SPI.WriteRead
Parameters:	Data, Integer (optional)
Returns:	Data
Description:	Writes a series of bytes to the SPI bus and returns the exact number of bytes read back. Using this command, you can execute any specific SPI op code. Writes data to the SPI bus and then optionally allows you to read data back. The first parameter is the data to write, it can be a hex string or data variable. The second parameter (which is optional) is the number of bytes to read back. Using this command you can execute any specific SPI op code. The example shows you how to read the SPI Flash's Device ID.
Examples:	chip_id = SPI.WriteRead(0x9F,3) print(chip_id)

JTAG commands

Command:	idcode
Returns	Integer
Description:	Returns the current IDCODE from the current selected JTAG device. Returns 0 if no JTAG device has been detected.
Examples:	dev_id = JTAG.idcode()

Command:	config
Parameters:	String
Description:	Configures the JTAG library to use processor specific instructions for the memory access.
Settings:	“MIPS” – Will load EJTAG extensions “ARM” – Will load ARM extensions
Examples:	JTAG.mode(“MIPS”) #For EJTAG devices

Command:	select
Parameters:	integer
Description:	Configures the JTAG library to use processor specific instructions for the memory access.
Examples:	JTAG.select(0) #Select the first device in a multi device chain

Command:	JTAG.Writeword
Parameters:	Integer, Integer
Description:	Writes a word of data (32-bit) to the JTAG device. First parameter is the 32-bit address and the second parameter is the word to write.
Examples:	JTAG.Writeword(0xFFFF1000,0xFF10)

Command:	JTAG.Readword
Parameters:	Integer
Returns:	Integer
Description:	Reads a word (32-bits) from the JTAG device and returns the value.
Examples:	RES = JTAG.Readword(0x1FC00000)

Command:	control
Parameters:	Integer

Returns:	Integer
Description:	Sends a JTAG control message to the target device. These types of commands are very dependent on the target device. This can be used to stop (0x10000) or start (0x0) the target processor. The result of the command is returned.
Examples:	JTAG.control(0x10000) #Stops the target processor

Command:	JTAG.MemoryInit
Parameters:	String, Integer, Integer (optional)
Description:	Initializes the JTAG memory controller and connects the host to a memory device connected to the target processor. First parameter is the memory type: “DMA”, “CFI”, or “SPI”. The second parameter is the DMA address (for DMA/CFI) or the SPI controller index: 1=Broadcom, 2=Atheros. The optional third parameter is the size of the memory devie (for DMA only).
Examples:	MemIndex = JTAG.MemoryInit(“CFI”,0x1FC00000) MemIndex = JTAG.MemoryInit(“SPI”,1) #BCM SPI

Command:	JTAG.Debug
Parameters:	Bool (true or flase)
Description:	Writes the JTAG data register with the standard flag to put the target device into debug mode: (`PRACC|PROBEN|SETDEV|JTAGBRK`)
Examples:	JTAG.Debug(true) #Will send the JTAG debug command

Command:	JTAG.CpuReset
Description:	Writes the JTAG data register with the standard flag to issue a processor reset. This command can have different results depending on the particular processor part: (`PRRST|PERRST`)
Examples:	JTAG.CpuReset #Will send a CPU reset command

Command:	JTAG.RunSVF
Parameters:	Data
Description:	This command will run a “Serial Vactor Format” file and process and write all of the commands to a connected JTAG device. This can be use to program Xilinx or Lattice CPLDs for example.
Examples:	JTAG.RunSVF(DataVar) #Runs a *.SVF file

Command:	JTAG.RunXSVF
Parameters:	Data
Description:	This command will run a compact (binary) “Serial Vactor Format” file and process and write all of the commands to a connected JTAG device. This can be use to program Xilinx CPLDs for example.

Examples:	JTAG.RunXSVF(DataVar) #Runs a *.XSVF file

Command:	JTAG.ExitState
Parameters:	Bool
Description:	This command will will enable or disable the go to test-logic-reset after a SVF file has been executed. Default behavior is enabled.
Examples:	JTAG.ExitState(False)

Command:	JTAG.ShiftDR
Parameters:	Data, Integer, Bool (optional)
Returns:	Data
Description:	Selects the DR register and then shifts data into it. First parameter is the data array, second parameter is the number of bits to shift in. The last optional parameter specifies to leave the DR-resister, the default is yes. The TDO data is shifted in and is returned.
Examples:	JTAG.ShiftDR(DATA,32)

Command:	JTAG.ShiftIR
Parameters:	Data, Integer, Bool (optional)
Returns:	Data
Description:	Selects the IR register and then shifts data into it. First parameter is the data array, second parameter is the number of bits to shift in. The last optional parameter specifies to leave the DR-resister, the default is yes. The TDO data is shifted in and is returned.
Examples:	JTAG.ShiftIR(DATA,32)

Command:	JTAG.ShiftOut
Parameters:	Data, Integer, Bool (optional)
Returns:	Data
Description:	Shifts data out at the current state. The first parameter is the data to shift, the second is the number of bits, the last parameter (optional) is to set the last TMS bit to exit the state (default is no)
Examples:	JTAG.ShiftOut(DATA,32,False)

Command:	JTAG.TapReset
Parameters:	None
Description:	Resets the test-access-port state machine to test-logic-reset.
Examples:	JTAG.TapReset()

Command:	JTAG.State
Parameters:	String
Description:	Changes the current state of the TAP. The parameter is a string. Valid inputs are: “RunTestIdle”, ”Select_DR”, ”Capture_DR”, ”Shift_DR”, ”Exit1_DR”, ”Pause_DR”, ”Exit2_DR”, ”Update_DR”, ”Select_IR”, ”Capture_IR”, ”Shift_IR”, “Exit1_IR”, “Pause_IR”, “Exit2_IR”, “Update_IR”.
Examples:	JTAG.State(“RunTestIdle”)

Command:	JTAG.GrayCode
Parameters:	Integer, Bool (optional)
Returns:	Integer
Description:	Returns the gray code (8-bit table) for a specific index. Optional parameter is used to specify if the table reverse should be used.
Examples:	JTAG.GrayCode(2,True) #Returns 0xC0

Boundary Scan Programmer commands

Command:	BoundaryScan.Init
Parameters:	None
Description:	Initializes the boundary scan module.
Examples:	BoundaryScan.Init()

Command:	BoundaryScan.AddPin
Parameters:	String, Integer, Integer, Integer (optional)
Returns:	Integer
Description:	Adds a pin to BSR association. This is how you define and map which pins from a Flash device should go to which pin on the target device. Parameter list: pin name, output/bidir index, control index, input index (opt) Valid pin names: DQx, ADx, WE#, OE#, CE#, WP#, RESET#, BYTE# Note: AD, DQ, WE and OE pins are mandatory, and the others are optional if the board does not route them to the host controller. If the IO cell is bidir and the control cell is next index, then you can omit the other 2 parameters.
Examples:	BoundaryScan.AddPin(”DQ1”, 331)

Command:	BoundaryScan.Detect
Parameters:	None
Description:	Initializes and attempts to detect a CFI compatible NOR memory as specified using the Setup and AddPin commands.

Examples:	BoundaryScan.Detect()

Miscellaneous commands

Command:	Writeline (or print)
Parameters:	Any data type
Description:	Displays a message to the console. You can input an Integer, String, or Data.
Examples:	writeline("this is only a test")

Command:	Msgbox
Parameters:	String
Description:	Displays a message to the user using a pop-up box.
Examples:	msgbox("Hello World!")

Command:	Status
Parameters:	String
Description:	This sets the status text (the bottom bar of the software).
Examples:	status("script is complete")

Command:	Refresh
Parameters:	None
Description:	Updates any of the connected memory device's hex editors. Useful if you have modified any of the screen contents in a script file.
Examples:	status()

Command:	Sleep
Parameters:	Integer
Description:	Waits the specified amount of time (in milliseconds), useful only in scripts.
Examples:	Sleep(1000) #Waits 1 second

Command:	Verify
Parameters:	Bool
Description:	Used to enable or disable the flash programming verification process.
Examples:	Verify(true)

Command:	Mode
Parameters:	None

Returns:	String
Description:	Returns the current operation mode: SPI, JTAG, EXTIO, I2C, etc.
Examples:	mode() #Returns "JTAG"

Command:	ask
Parameters:	String
Returns:	Bool
Description:	Asks the user a yes or no question and returns that value. You can use this in an if statement to make conditional sections.
Examples:	ask("Continue script?")

Command:	endian
Parameters:	String
Description:	Allows the endian mode to be changed. The input can be a string to set the mode: “MSB” or “LSB”. Some JTAG devices will need to have this setting specified. “MSB” “LSB16” “LSB8” are valid options.
Examples:	endian(“LSB”)

Command:	abort
Description:	Aborts any script that is running. Use this in the console to quit a script that may not be working as intended.
Examples:	abort

Command:	crc32
Parameters:	Data
Returns:	Integer
Description:	Computes a standard CRC-32 checksum for a given data variable. The command crc16 is also supported for legacy uses.

Examples:	data_var = memory.read(0,memory.size) #Reads all data from a device
	c32_value = crc32(data_var) #Generates checksum
	print(string.hex(c32_value)) #Prints the hex value