BlackcatUSB Script Engine

This document is to introduce you to the powerful and flexible script engine that is built into BlackcatUSB. The purpose of this feature is to allow you to create device specific scripts for manipulating the data on the flash.

The BlackcatUSB script engine is modeled after the vbscript language for efficiency and simplicity.

How a script file is executed

A script file can be executed one of two ways, automatically (like in the case of JTAG) or manually from the "Load Script" button on the front tab.

In JTAG mode, when a device is detected, the device sends BlackcatUSB its manufacture ID and part number. The script engine then looks for the name of that ID in the script folder and if found will run that script. If it was not found, the script engine will instead run the "default.bcs" script.

In SPI mode, only flash information is retrieved, so the software will not know what device it is being used in or what it is for. Therefore, you will need to load the script manually after the software has detected and connected to the Flash memory.

Script file structure

A script file is just a plain text document. You can open and edit one using Notepad. The contents of a script file is made up of commands, control tags, and events. When a script file is ran, any non event command will be executed.

Events

A event is a series of commands that are executed together, much like a function. Events can only be executed two ways, by either using its name as a function or from assigning it to a button.

To create a event, you use CreateEvent(EventName) where EventName is the name you will specify. At the end of your event you must be a EndEvent tag to let the script engine know the event is over.

Events are very useful. You can pass variables to events and retrieve values from events. When you pass a variable or value to a 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.

For example a script that looks like:

```
JiggaJigga("Hello World")

CreateEvent(JiggaJigga)
   msgbox($1)
EndEvent
```

Will popup "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(JiggaJigga("Hello"," World"))

CreateEvent(JiggaJigga)
    StrVar = $1 & $2
    Return StrVar
EndEvent
```

Commands

Commands are built in functions that you will want to use to access the functionality of the software. Some functions can be used to retrieve values and some are used only to do certain tasks. You can test out commands by entering them into the software's console page. This can be a good way to test out elements of your script in real time, without the need to close and restart the software each time. For a list of all the commands you can use, see Script_Commands.pdf

Variables

A variable is a name that you assign a object too. You can assign a string, data, integers, or boolean values.

```
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

Adding Variables

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.

Conditions

Simply put, you can create a IF ELSE statement to execute code based on a condition. Use the tag "If" followed by a condition statement. You can add a "else" tag to execute if the statement is evaluated false. End the condition using the tag "EndIf"

For example, take the following code:

```
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 functions that return TRUE or FALSE in a If statement.

If you precede the condition statement with the "not" keyword, what ever the statement is evaluated at, the opposite will happen.

```
If not (GetValue() > 10)
    msgbox("This will be executed")
EndIf

CreateEvent(GetValue)
    retVar = 5
    return retVar
EndEvent
```

Reading or writing to memory (including flash devices)

BlackcatUSB allows you to create connections from the software to the flash device on a target system. Since a target system can contain multiple memory devices, you can create individual tabs and scripting variables for each device.

```
JTAG.MemoryAddress(0x0)
JTAG.MemoryType("RAM")
JTAG.MemorySize(0x800000)
CFGMEM = JTAG.MemoryInit()
```

The above scripting code shows you how to create a single memory device. The first line sets the physical address of where the memory is located on the DRAM chain. This can change from device to device.

The second line sets the memory type, this can be "RAM" or volatile memory, "CFI" for non-volatile, or "SPI" for serial memory.

The third line sets the size of the device. This is only required for volatile memory such as RAM. The size of CFI and SPI devices will automatically be discovered by the Init.

The last command is the Init function. It will make BlackcatUSB perform all the functions to create the memory device. This function can also be used to store the flash index to a variable. Each time you successfully create a flash device, it will return the unique index of the device. So the first device will be index 0, the second index 1 and so on.

Now that you have created a memory device, refer to the Script_Functions.pdf for all of the sub functions you can use with a memory device.

Script Control

To allow you to control the execution of the script, you can use many built in tags.

The tag "goto" can be used to change the current position of your script that is executing. To do so, create a label by creating a name and ending it with ":" then when your script executes "goto <that label name>" the script will go to that position.

The tag "exit" can be used to leave the current section of code. If this is used in a If statement, the script position will jump to the next command after the EndIf tag. If you want to leave a event, you can specific "exit event" or if you want to quit the whole script all together, use "exit script".

Autorun Feature (JTAG mode only)

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:
<CPU_ID>:<SCRIPT_NAME>:<DEVICE_NAME>

Add as many scripts as you need and when you run the software, when it connects via JTAG it will load all of the scripts that match the CPU ID. Then the first device will be selected on the GUI and executed. 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.