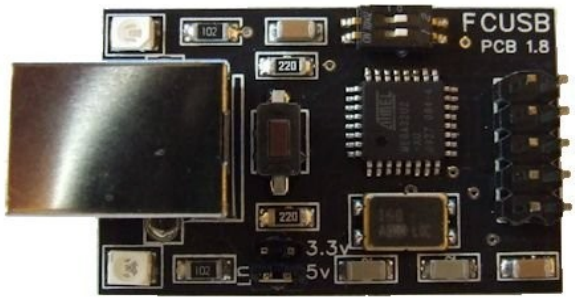
Embedded Computers

For professionals



FlashcatUSB

USER'S GUIDE

Release Candidate 15

Website: www.embeddedcomputers.net/products/FlashcatUSB/
Support email: contact@embeddedcomputers.net
Last updated: April, 2012

FLASHCAT

THE INDEX

INTRODUCTION

The FlashcatUSB device is a versatile multi-protocol Flash memory programmer. It is a cost effective hardware tool that is used both by industry professionals and hobbyists. By utilizing a single USB micro-controller design, the hardware in conjunction with software and firmware can allow the device to configure its operation to meet the required protocols of a desired TAP (test-access point) or interface bus (such as a serial peripheral interface). Since the device uses a hardware based USB interface, it can then perform its function at a much faster rate than traditional serial or parallel ports.

The hardware board features:
USB 1.1 and 2.0 compatible
32KB of programmable memory (4KB reserved for the USB bootloader)
16 MHz RISC based-processor (Atmel AVR 8-bit core)
Two switches for application mode changes
Output voltage selector jumper (3.3v or 5v)
LED for indicating current application status
Button for resetting the device or for activating the bootloader

Currently supported features for JTAG mode:
CFI compatible flash memory – Intel, AMD, and SST algorithms supported
DMA and PrAcc modes supported for target memory access
MIPS supported (for EJTAG)
Instruction register (IR) auto-sected from 4 to 512 bits.

Currently supported features for SPI mode:
Mode 0, 1, and 2 compatible
High density devices supported: 1 to 128 mbit.
High-speed mode (FOSC/2) – Reads up to 400KB per second
Ability to program MCU's with on board Flash / NV memory

SOFTWARE REQUIREMENTS

A computer with at least a 1 GHz processor and 256 MB of free memory available.

Operating systems supported: Windows XP, Windows Vista, Windows 7. Supports both 32-bit and 64-bit environments. Apple OS X and Ubuntu version is coming soon.

Microsoft .net Framework 4.0:
Download: http://www.microsoft.com/download/en/details.aspx?id=17851 or from Windows Update

LIST OF SUPPORTED FLASH DEVICES

This is only a partial list of devices that are supported, as the CFI mode can automatically configure to any device that is detected, and in SPI mode the user can self-configure the device needed to be programmed.

Verified CFI compatible flash devices:

Spansion S29GL256M	AMD 29LV320DT	Intel TE28F320C3T
Spansion S29GL128M	AMD 29LV320MB	Intel TE28F320C3B
Spansion S29GL064M	AMD 29LV320MT	Intel TE28F640C3T
Spansion S29GL064M	AMD 29LV400BB	Intel TE28F640C3B
Spansion S29GL032M	AMD 29LV800BB	Intel 28F320J5
Spansion S70GL02G	ATMEL AT49BV/LV16X	Intel 28F640J5
Spansion S29GL01G	ATMEL AT49BV/LV16XT	Intel 28F320J3
Spansion S29GL512	HYHYNIX HY29F400TT	Intel 28F640J3
Spansion S29GL256	HYHYNIX HY29LV1600T	Intel 28F128J3
Spansion S29GL128	Intel 28F160B3	Samsung K8D1716UB
AMD 28F400BT	Intel 28F160B3	Samsung K8D1716UT
AMD 29DL322GB	Intel 28F800B3	Samsung K8D3216UB
AMD 29DL322GT	Intel 28F320B3	Samsung K8D3216UT
AMD 29DL323GB	Intel 28F320B3	ST M28W160CB
AMD 29DL323GT	Intel 28F640B3	ST M29D323DB
AMD 29DL324GB	Intel 28F640B3	FUJITSU 29DL323GB
AMD 29DL324GT	Intel TE28F800C3T	FUJITSU 29DL323TE
AMD 29LV160DB	Intel TE28F800C3B	FUJITSU 29LV160B
AMD 29LV160DT	Intel TE28F160C3T	FUJITSU 29LV160T
AMD 29LV320DB	Intel TE28F160C3B	FUJITSU 29LV320BE
Micron 28F160C34B	MXIC 25FL0165A	FUJITSU 29LV320TE
Micron 28F160C34T	MXIC 29LV800T	FUJITSU 29LV800B
Micron 28F322P3	MXIC 29LV800B	TOSHIBA TC58FVT160B
SHARP 28F320BJE	MXIC 29LV161B	TOSHIBA TC58FVB321
SHARP LH28F160BJHG	MXIC 29LV161T	TOSHIBA TC58FVT160
SHARP 28F160S3	MXIC 29LV320B	TOSHIBA TC58FVT321
SHARP 28F320S3	MXIC 29LV320T	SST 39VF1600
ST MT28W320	MXIC 29LV800BMC	SST 39VF1601
ST 29W320DB	ST M58LW064D	SST 39VF3201
ST 29W320DT	ST M29W800AB	SST 39VF800
ST M29W160EB	ST M29W160ET	

Verified SPI compatible flash devices:

Atmel AT25DF641	Windbond W25X40	EON EN25P16
Atmel AT25DF321	Windbond W25X80	EON EN25P32
Atmel AT25DF161	Windbond W25X16	EON EN25P64
Atmel AT25DF081	Windbond W25X32	SST 25VF010A
Atmel AT26DF081A	Windbond W25X64	SST 25VF020A
Atmel AT26DF161	Windbond W25Q80BV	SST 25WF040
Atmel AT26DF161A	Windbond W25Q16BV	SST 25VF040B
Atmel AT26DF321	Windbond W25Q32BV	SST 25VF080
Spansion S25FL128P	Windbond W25Q64BV	SST 25VF080B
Spansion S25FL064	MXIC MX25L10	SST 26VF016
Spansion S25FL032	MXIC MX25L20	SST 25VF016B
Spansion S25FL016	MXIC MX25L40	SST 25VF032
Spansion S25FL008	MXIC MX25L80	SST 25VF032B
ST M25P128	MXIC MX25L160	SST 25VF064B
ST M25P64	MXIC MX25L320	SST 26VF064
ST M25P32	MXIC MX25L640	PCT 25VF064C
ST M25P16	MXIC MX25L128	SST 25VF128B
ST M25P80	MXIC MX25L4006	PMC PM25LV010
ST M25P40	EON EN25F20	PMC PM25LV020
ST M25P20	EON EN25F40	PMC PM25LV040
ST M25P10	EON EN25F80	PMC PM25LV080B
Atmel AT45DB011	Atmel AT45DB321	PMC PM25LV080B
Atmel AT45DB021	Atmel AT45DB642	Atmel AT45DB081D
Atmel AT45DB041	Atmel AT45DB011D	Atmel AT45DB161D
Atmel AT45DB081	Atmel AT45DB021D	Atmel AT45DB321D
Atmel AT45DB161	Atmel AT45DB041D	Atmel AT45DB642D

Support MCU's with on board memory programmable via SPI

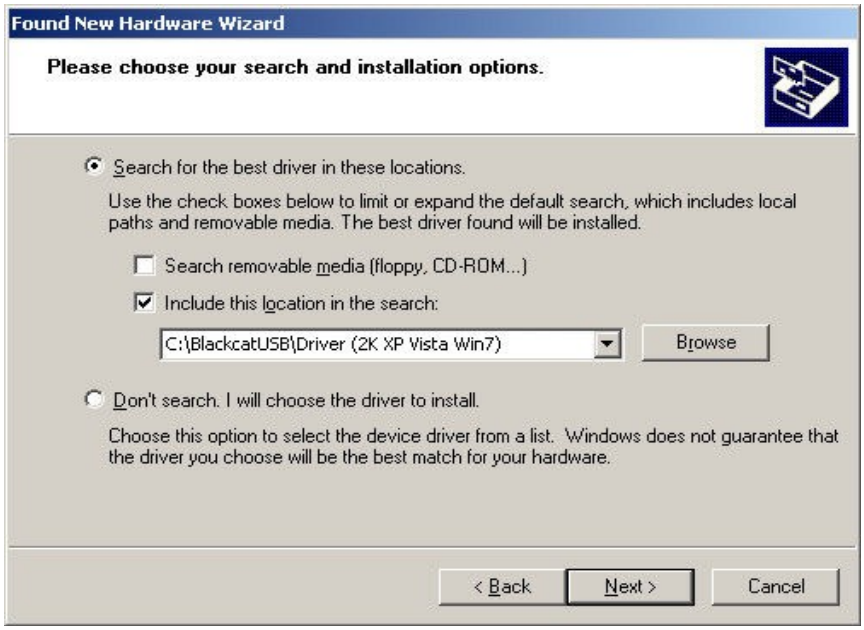
Nordic nRF24LE1

If your flash is not listed above, just contact us. We can add almost any flash device upon request. We do this frequently for many companies and individuals.

DRIVER INSTALLATION

When you connect FlashcatUSB to your PC for the first time, you should notice a "Found new hardware" notification pop-up, after which this prompt should appear:

Select "Install from a list or specific location" and click Next.



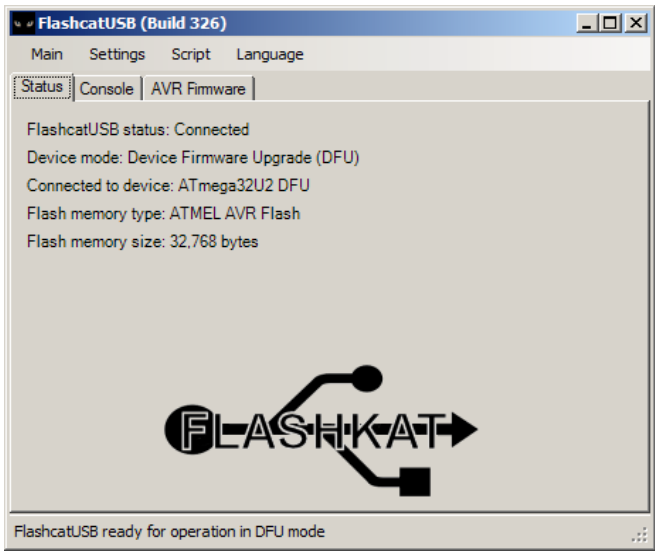
Click Browse and locate the driver folder from the software package and click Next to install the software drivers. You will need to repeat this process once for each AVR firmware you use.

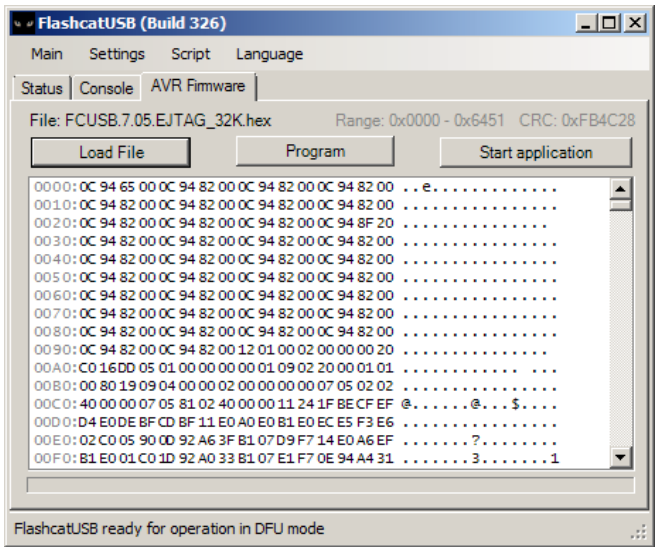
SETTING UP DEVICE FOR APPLICATION MODE

FlashcatUSB is shipped in bootloader mode. That is a mode (also known as the DFU mode) which allows the software to perform a unit firmware update over USB. The functionality of this device differs depending on the type of firmware programmed into it. Once programmed with a specific firmware, you can easily put the device back into DFU mode by putting the device into “Bootloader Mode” and then pressing the reset button on the hardware.

Application Mode

Bootloader Mode



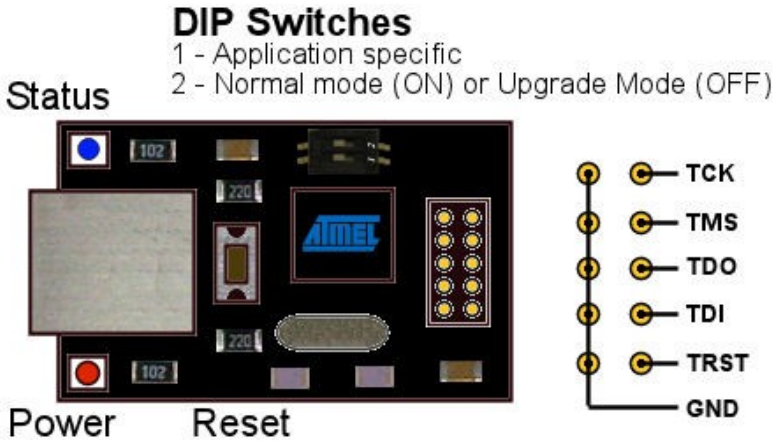


In “Application Mode”, pressing the hardware button will only perform a USB reset. In “Bootloader Mode”, it will reset the device but when the device automatically reconnects, it will start in the DFU state, allowing the software to load new AVR firmware.

To load new firmware, go to the “AVR Firmware” tab, click the “Load File” button, select the firmware from inside the folder that accompanies the software, click “Program” and finally “Start Application” to reset the device which will cause it to begin the application from the firmware you selected.

USING JTAG MODE FOR FLASH PROGRAMMING

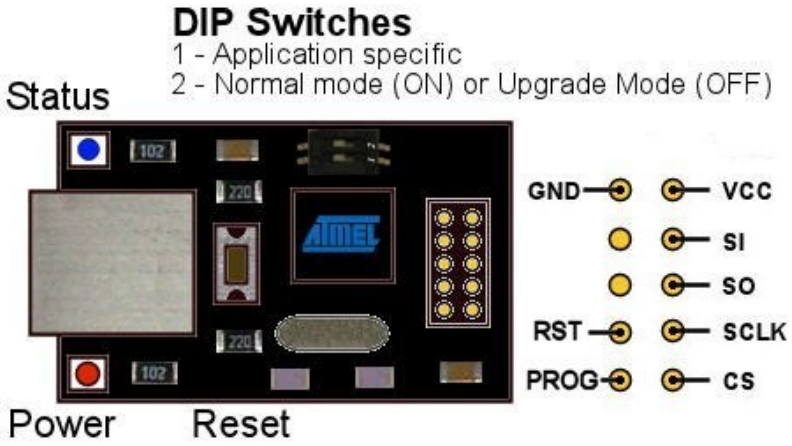
With the “EJTAG” version of the AVR Firmware installed, you can use FlashcatUSB to interface with a SoC (system-on-a-chip) processor over JTAG. This allows the software to then integrate and communicate directly with attached memory devices, usually memory (in the form of DRAM) and storage in the way of non-volatile Flash memory.



The image above shows you the pin outs of the 10-pin port and how it should be connected to the test-access port (TAP) of your target device. FlashcatUSB will only act like a passive diagnostic tool for the target system, so the device will need to be powered on by itself.

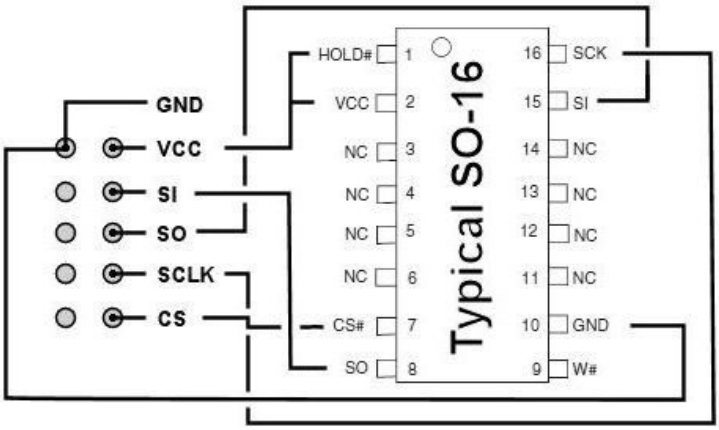
USING SPI MODE FOR FLASH PROGRAMMING

With the “SPI” firmware installed, you can use the device as a high-speed programmer for virtually every SPI compatible Flash memory.

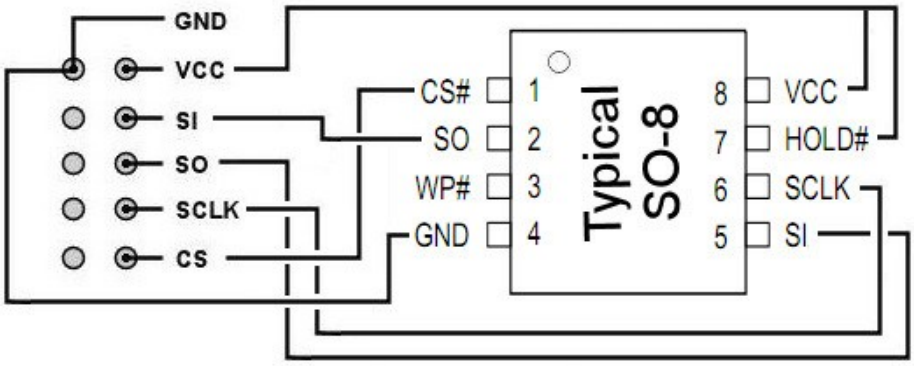


The image above shows you the pin outs of the 10-pin port and how it should be connected to the SPI bus of your targeted device. Unlike JTAG mode, you can use the VCC pin to power the target device. You should make note of the external power the chip needs and to make sure you have that voltage selected on the FlashcatUSB board. Most SPI chips use the 3.3v setting.

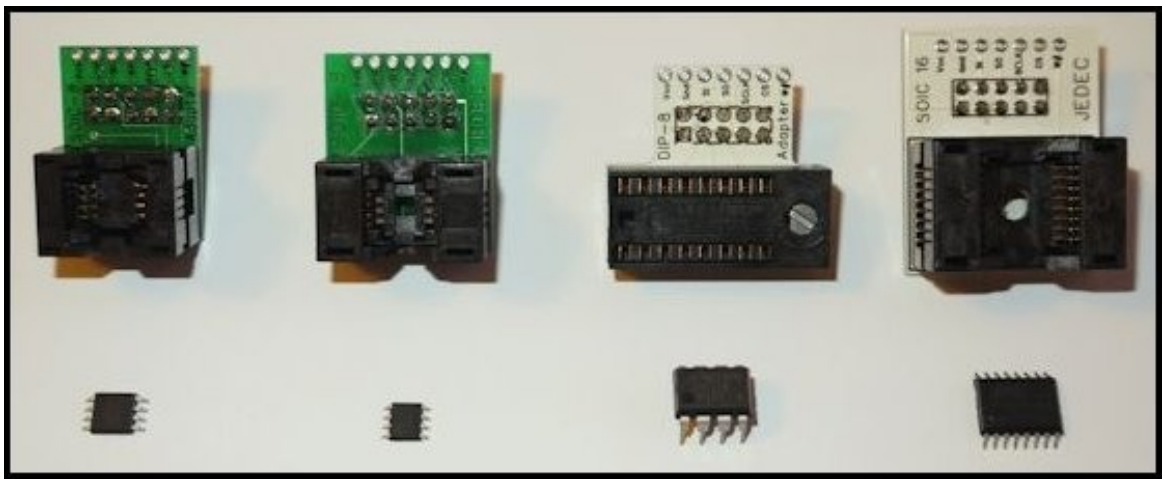
*__Note__: the RST and PROG pins are only used when connecting to MCU's with on board flash, such as the Nordic nRF24LE1 device.



The above diagram is how you should connect the 10-pin port to a typical SOIC-16 package of a SPI compatible Flash device.

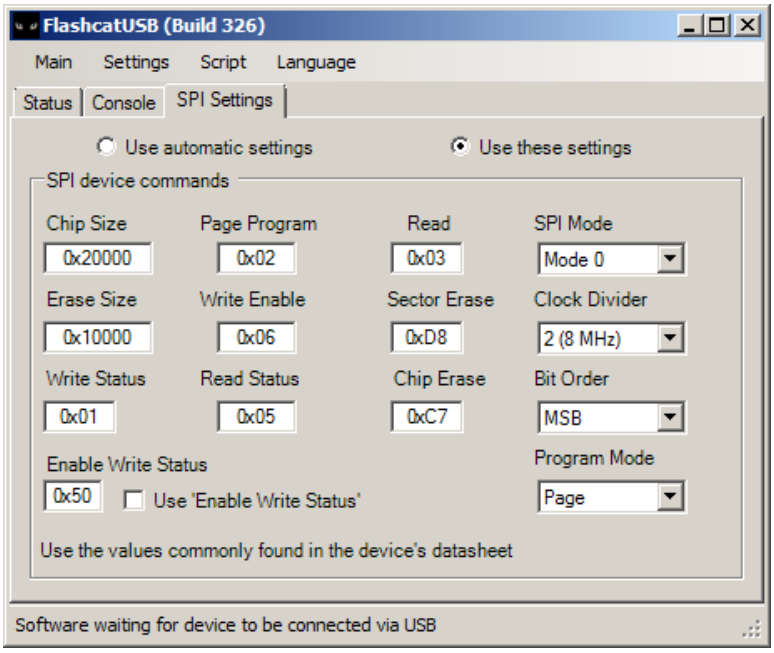


The above diagram is how you should connect the 10-pin port to a typical SOIC-8 and DIP-8 package of a SPI compatible Flash device.



Optionally, you can also purchase drop-in sockets for most common SPI chip packages.

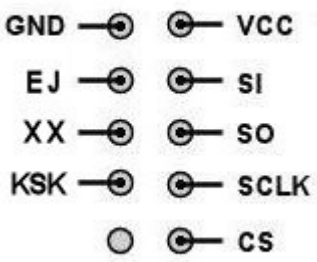
When you start FlashcatUSB with the SPI device connected, the software should automatically detect the chip and load its settings accordingly. If the chip is detected, but not supported by the software, you can then use the SPI Manual Settings tab to specify all of the settings with the values from the chip's datasheet.



USING NAND MODE FOR FLASH PROGRAMMING

Ultra-high density non-volatile memory chips (usually in the gigabits) come in a parallel package (usually BGA) that are connected to a common shared bus (that is wired for JTAG or SPI) and you can also use FlashcatUSB to program those types of chips.

To use this feature, connect the 10-pin port on FlashcatUSB to the target device's SPI bus.



Pin-out for NAND Mode

GETTING STARTED WITH THE SCRIPT ENGINE

One of the most powerful features of FlashcatUSB is a customizable scripting engine. This can help engineers speed up many common tasks. This engine uses text based files that is comprised of a language that is simiar to vbscript. This makes the script feature easy to learn and very efficient for manipulating data.

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" menu from the software.

In JTAG mode, when a device is detected, the device sends FlashcatUSB its manufacture ID and part number. The software will then check the file “Autorun.ini” for that ID and if it finds it, it will load the corresponding script specified. If the ID is not found, then the “default.bcs” script file is executed.

In SPI mode, you can manually run a script from the “Script” menu.

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, labels, 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 an 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.

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
```

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. 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
```

Creating access to a memory device

FlashcatUSB 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 FlashcatUSB 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.

Script Control

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

GOTO
The "goto" keyword can be used to change the current position in your script that you are

executing. The syntax is 'goto: <label>'. A label is a simple tag name that you can place anywhere in your current event. To create one, make a name and end it with a colon. Using GOTO keywords, you can also create simple loops.

```
VarInt = 0                    #This creates a variable and gives it a value of 0
TOPCODE:                      #This creates the label, named TOPCODE
If (VarInt = 10)              #This does a compare to see if VarInt is 10
   GOTO TOPCODEDONE  #This keyword will make a jump to the end of the code
Endif
VarInt += 1
GOTO TOPCODE              #This will always jump to the top
TOPCODEDONE:               #When your code is done, it jumps here
```

EXIT

The "exit" keyword can be used to skip a section of code, to exit out of a current condition segment (i.e. if/endif or while loop). The syntax is 'exit' or 'exit <event>' or 'exit script'. If you run exit script, no matter what event you are in, the entire script will stop execting.

For example, if you are in a condition statement and want to leave, you can do this:

```
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
```

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.

LIST OF CONSOLE OR SCRIPT FUNCTIONS

The following is a list of commands that are built into the application. You can access these commands from both the scripting engine (via device scripts) and the console tab of the application.

Some of these commands are specific to the mode in which FlashcatUSB is connected, these are namely the commands that begin with SPI or JTAG.

The Memory commands are index. That is, if you have created more than one memory device, specifying the index will allow you to access that specific device. 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. When you create a memory device using the JTAG.MemoryInit, you should save that index to a variable so you can specify which device to perform the memory operation from.

Some of the functions can return values. These values can be put into variables for further processing. The types of the values are:

Bool - is a value of either TRUE or FALSE

String - is a value of readable characters. This value is indicated with surrounding quotes

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.

Command:	SetParam
Parameters:	Integer, Integer
Syntax:	Setting, Value
Description:	Sets a device parameter on the board (firmware controlled). The delays are set in milliseconds and is the amount of time the AVR should wait between read or write instructions. The main purpose of this command is to fine tune performance; the faster the device operates, the higher the error rate is. This can also affect different target devices differently.
Settings:	1: Intel Flash delay 2: AMD Flash delay 3: Memory read delay
Examples:	SetParam(1, 30) #Sets the Intel flash delay to 30 ms

Command:	Ejctrl
Parameters:	Integer
Returns:	Integer
Description:	Sends a JTAG control message to the target device. These types of commands are very dependant 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:	Ejctrl(0x10000) #Stops the target processor

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

Command:	FixFlash
Parameters:	None
Description:	Attempts to reprogram the bootloader of a device blindly (no verification, no check device id etc.). This is sometimes successful in restoring a device that does not boot correctly. Only supported in JTAG mode.
Examples:	FixFlash()

Command:	Verify
Parameters:	None or Bool
Returns:	Bool or nothing
Description:	Used to enable or disable the flash verify process. It can also be used to return the current setting.
Examples:	Verify(true)

Command:	writeline
Parameters:	String
Description:	Displays a message to the console.
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:	mode
Parameters:	None
Returns:	String
Description:	Returns a string indicating which mode FlashcatUSB is in.
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 a if statement to make conditional sections.

Examples:	ask("Continue script?")

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

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","Jigga Jigga!")

Command:	Tab.ButtonDisable
Parameters:	String
Description:	Disables a button so the user can not click it and run the event.
Examples:	Tab.ButtonDisable("btName")

Command:	Tab.ButtonEnable
Parameters:	String
Description:	Enables the button (if you had it disabled)
Examples:	Tab.ButtonEnable("btName")

Command:	JTAG.MemoryAddress
Parameters:	Integer
Description:	Initialized the dynamic memory controller and sets the base memory address.
Examples:	JTAG.MemoryAddress(0x80000000)

Command:	JTAG.MemoryType
Parameters:	String
Description:	Sets the device type of the memory. This can be "RAM", "CFI" or "SPI". Note: SPI mode over JTAG is not yet supported.
Examples:	JTAG.MemoryType("CFI")

Command:	JTAG.MemorySize

Parameters:	Integer
Description:	Sets the size of the memory (in bytes) of the dynamic memory
Examples:	JTAG.MemorySize(0x800000)

Command:	JTAG.MemoryInit
Parameters:	None
Description:	Will initialize and connect the FlashcatUSB interface to the memory interface. You may need to specify address and size prior to calling this function. If successful, the GUI will add the "Memory" tab. This command also returns the unique index of the created device.
Examples:	MemIndex = JTAG.MemoryInit()

Command:	JTAG.FlashInit
Parameters:	None
Description:	Will connect to the CFI compliant flash on the memory controller to allow for reading and writing. This will create the "Flash" tab on the GUI. Must set FlashBase prior.
Examples:	JTAG.FlashInit()

Command:	JTAG.FlashFind
Parameters:	None
Description:	Will scan the entire memory address range for a CFI compatible flash.
Examples:	JTAG.FlashFind()

Command:	JTAG.BigEndian
Parameters:	None
Description:	Sets the endian for the JTAG memory devices to big.
Examples:	JTAG.BigEndian()

Command:	JTAG.LittleEndian
Parameters:	None
Description:	Sets the endian for the JTAG memory devices to little.
Examples:	JTAG.LittleEndian()

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.Reset
Syntax:	
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.Reset #Will send a JTAG reset command

Command:	JTAG.EnableVCC
Parameters:	Bool (true or flase)
Description:	Enables the JTAG VCC pin to output voltage. This can be either 5v or 3.3v depending on the board jumper. This pin can provide up to 100ma of power and can be suitable for operating most memory devices, although not enough to power an entire target board.
Examples:	JTAG.EnableVCC(true) #Will make the VCC pin output power

Command:	JTAG.RunSVF
Parameters:	Data (byte array)
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 CPLDs for example.
Examples:	JTAG.RunSVF(DataVar) #Runs a *.SVF file

Command:	JTAG.RunXSVF
Parameters:	Data (byte array)
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:	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. Works for both CFI and SPI type flash

	devices, but please note you must have already initiated the flash.
Examples:	dataVar = Memory.Read(0,512) #Reads 512 bytes

Command:	Memory.ReadString
Parameters:	Integer
Returns:	String
Description:	Reads a string from the location 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.GetSectorCount
Returns:	Integer
Description:	Erases the specified flash sector
Examples:	sectors = Memory.GetSectorCount()

Command:	Memory.EraseSector
Parameters:	Integer
Returns:	Nothing
Description:	Erases the specified flash sector
Examples:	Memory.EraseSector(0)

Command:	Memory.EraseSection
Parameters:	Integer, Integer
Returns:	Nothing
Description:	Erases a section of the flash memory, starting at the address (the first parameter), and the number of bytes (second parameter).
Examples:	Memory.EraseSector(0x10000,0x8000)

Command:	Memory.EraseBulk
Parameters:	None
Returns:	Nothing

Description:	Erases the entire flash memory
Examples:	Memory.EraseBulk()

Command:	Memory.GetSectorSize
Parameters:	Integer
Returns:	Integer
Description:	Returns the size
Examples:	dataInt = Memory.GetSectorSize(0)

Command:	Memory.Backup
Parameters:	None
Description:	Previously known as "Dump", this reads all the data from flash (twice) and then prompts the user to save the file to disk. Usefully for making a flash backup that has data integrity.
Examples:	Memory.Backup()

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()

Command:	Memory.WriteWord
Parameters:	Integer, Integer
Description:	Writes an integer (all 32 bits) to a specific address on the memory device.
Examples:	Memory(2).WriteWord(0x80010000,0x32)

Command:	SPI.Fosc
Parameters:	Integer
Description:	Used to set the hardware SPI clock divider. The SPI speed is the system clock (16 MHz) divided by the Fosc value. Supported values: 2, 4, 8, 16, 32, 64, 128
Examples:	SPI.Fosc(4)

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.Swap
Parameters:	Bool
Description:	Used to reverse the bit order of bytes of the data being written or read to the flash. For example, if your flash uses MSB, but your microprocessor is LSB and reads the data off of the SPI flash, you can use this command to conveniently swap the bits.
Examples:	SPI.Swap(true)

Command:	OpenFile
Parameters:	String, String (optional)
Returns:	Data
Description:	Prompts the user for a file and then reads the file from disk and returns a data variable. First parameter is the title of the window and the optional second is the standard file filter to use.
Examples:	MyData = OpenFile("Choose file", "Firmware files (*.bin)|*.bin")

Command:	SaveFile
Parameters:	Data, String, 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:	SaveFile(MyData,"Where to save?","fileread.bin")

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

Command:	resize
Parameters:	Data, Integer, Integer (optional)
Description:	Resizes a byte array (usually in a variable), starting at the first parameter. The optional parameter can be used to specify how many to copy.
Examples:	resize(datavar,2) #removes the first two bytes

Command:	Len

Parameters:	String, Integer or Data
Returns:	Integer
Description:	Returns the length of a string , the number of bytes in a data object or the number of bytes to hold a integer.

Examples:	len("hello") #returns 5
	dataVar = 0x1F;0x2F;0x2F;0xFF;0x2A;0x50
	len(dataVar) #returns 6
	len(302) #returns 2

Command:	Byte
Parameters:	Data, Integer
Returns:	Integer
Description:	Returns the value of the byte located in a data array.
Examples:	dataVar = 0x1F;0x3F;0x2F;0xFF;0x2A;0x50 word(dataVar,2) #Returns 47

Command:	Word
Parameters:	Data, Integer
Returns:	Integer
Description:	Returns the value of the four bytes located in a data array.
Examples:	dataVar = 0x1F;0x3F;0x2F;0xFF;0x2A;0x50 word(dataVar,2) #Returns 805251664

Command:	HWord
Parameters:	Data, Integer
Returns:	Integer
Description:	Returns the value of the two bytes located in a data array.
Examples:	dataVar = 0x1F;0x3F;0x2F;0xFF;0x2A;0x50 hword(dataVar,2) #Returns 12287

Command:	value
Parameters:	String
Returns:	String
Description:	Returns the text value of a Tab object, such as a textbox you have created.
Examples:	strVar = value(MYINPUTBOX)

Command:	ToInteger
Parameters:	String
Returns:	Integer
Description:	Converts a integer stored in a string as a integer.

Examples:	ToInteger("234") #Returns 234

Command:	sigmakey
Parameters:	None
Returns:	String
Description:	Returns the SIGMA key of a given device. Only compatible in JTAG mode. Also prints the key to the console.
Examples:	strVar = sigmakey()

Command:	copy
Parameters:	Data, Data
Returns:	Data
Description:	Copies two data variables and returns them as a new combined data variable.
Examples:	dataVar = copy(data1,data2) #equals data1+data2