Embedded Computers

For professionals

FlashcatUSB

USER'S GUIDE

Release Candidate 19

Website: www.embeddedcomputers.net/products/FlashcatUSB/
Support email: contact@embeddedcomputers.net
Last updated: October, 2016

THE INDEX

WHATS NEW IN RC19

1. Datasheet has been updated.
2. Includes AVR firmware 4.05 for PCB 2.x boards.
3. SPI mode now supports SQI (Duad I/O) protocol.
4. New console command: **spi.database** (prints out the supported devices)
5. Additional SPI devices and SPI EEPROMS have been added.

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, 2.0, and 3.0 compatible
32KB of programmable memory (4KB reserved for the USB bootloader)
16 MHz RISC based-processor (Atmel ATmega AVR 8-bit core)
Two switches: one for application and one for bootloader mode
Triple voltage output: 1.8v, 3.3v and 5.0v (current @ 120ma)
LED for indicating current application status
Button for resetting the device or for activating the USB bootloader (DFU compliant)

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
SQI protocol supported (26 series or other quad I/O devices)
High density devices supported: 1 to 128 mbit. (24-bit addressing)
Ultra-high density devices supported: 256 mbit to 1 Gbit (32-bit addressing)
High-speed mode (FOSC/2) – Reads up to 400KBytes per second
Ability to program MCU's with on board Flash / NV memory

Currently supported features for I2C mode:
Supports all I2C and TWI memory devices
Address selectable (A0, A1, A2)
Supports up to 400kHz data rates

SOFTWARE REQUIREMENTS

A computer with at least a 1 GHz processor and 256 MB of free memory available, and a USB 1.1 / 2.0 port. USB 3.0 is compatible with JTAG/SPI/NAND firmware, but not the manufacturer supplied device bootloader. This means you may need to change AVR firmware using a native USB 2.0 port.

Operating systems supported: Windows XP, Windows Vista, Windows 7, 8, 8.1, and 10.
Supports both 32-bit and 64-bit versions.
Microsoft .NET Framework 4.0 (Download)

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:

SST39LF020 (2Mbit)	Intel 28F128J3 (128Mbit)	MX25FL0165A (16Mbit)
SST39LF400A (4Mbit)	Intel 28F160B3 (16Mbit)	MX29LV800xT (8Mbit)
SST39LF401C (4Mbit)	Intel 28F160B3 (16Mbit)	MX29LV800xB (8Mbit)
SST39LF402C (4Mbit)	Intel 28F160C3B (16Mbit)	MX29LV160xT (16Mbit)
SST39LF800A (8Mbit)	Intel 28F160C3T (16Mbit)	MX29LV160xB (16Mbit)
SST39LF801C (8Mbit)	Intel 28F320B3 (32Mbit)	MX29LV320CT (32Mbit)
SST39LF802C (8Mbit)	Intel 28F320B3 (32Mbit)	MX29LV320CB (32Mbit)
SST39SF010A (1Mbit)	Intel 28F320C3 (32Mbit)	MX29LV640MT (64Mbit)
SST39SF020A (2Mbit)	Intel 28F320C3 (32Mbit)	MX29LV640MB (64Mbit)
SST39SF040 (4Mbit)	Intel 28F320J3 (32Mbit)	TC58FVT160 (16Mbit)
SST39VF010 (1Mbit)	Intel 28F320J5 (32Mbit)	TC58FVB321 (32Mbit)
SST39VF020 (2Mbit)	Intel 28F640B3 (64Mbit)	TC58FVB160 (16Mbit)
SST39VF040 (4Mbit)	Intel 28F640B3 (64Mbit)	TC58FVT321 (32Mbit)
SST39VF1601 (16Mbit)	Intel 28F640C3 (64Mbit)	TC58FVM6T2AFT65 (64Mbit)
SST39VF1601C (16Mbit)	Intel 28F640C3 (64Mbit)	TC58FVM6B2AFT65 (64Mbit)
SST39VF1602 (16Mbit)	Intel 28F640J3 (64Mbit)	TC58FVM6T2AXB65 (64Mbit)
SST39VF1681 (16Mbit)	Intel 28F640J5 (64Mbit)	TC58FVM6B2AXB65 (64Mbit)
SST39VF1682 (16Mbit)	Intel 28F800B3 (8Mbit)	M29W800AB (8Mbit)
SST39VF200A (2Mbit)	Intel 28F800C3 (8Mbit)	M29W160DT (16Mbit)
SST39VF3201 (32Mbit)	AMD S29GL320 (32Mbit)	M29W160DB (16Mbit)
SST39VF3201B (32Mbit)	AMD 28F400BT (4Mbit)	M29W320DT (32Mbit)
SST39VF3201C (32Mbit)	AMD 29DL322GB (32Mbit)	M29W320DB (32Mbit)
SST39VF3202 (32Mbit)	AMD 29DL322GT (32Mbit)	M28W320DT (32Mbit)
SST39VF3202B (32Mbit)	AMD 29DL323GB (32Mbit)	M28W320DB (32Mbit)
SST39VF3202C (32Mbit)	AMD 29DL323GT (32Mbit)	M28W640DT (64Mbit)
SST39VF400A (4Mbit)	AMD 29DL324GB (32Mbit)	M28W640DB (64Mbit)
SST39VF401C (4Mbit)	AMD 29DL324GT (32Mbit)	M58LW064D (64Mbit)
SST39VF402C (4Mbit)	AMD 29LV160DB (16Mbit)	M30LW128 (128Mbit)
SST39VF6401B (64Mbit)	AMD 29LV160DT (16Mbit)	K8D3216UT (32Mbit)
SST39VF6402B (64Mbit)	AMD 29LV320DB (32Mbit)	K8D3216UB (32Mbit)
SST39VF800A (8Mbit)	AMD 29LV320DT (32Mbit)	K8D3316UT (32Mbit)
SST39VF801C (8Mbit)	AMD 29LV320MB (32Mbit)	K8D3316UB (32Mbit)
SST39VF802C (8Mbit)	AMD 29LV320MT (32Mbit)	K8D3216UT (32Mbit)
SST39WF1601 (16Mbit)	AMD 29LV400BB (4Mbit)	K8D3216UB (32Mbit)
SST39WF800B (8Mbit)	AMD 29LV800BB (8Mbit)	K8D1716UT (16Mbit)

MBM29LV800TA (8Mbit)   S29AL008D (8Mbit)   K8D1716UB (16Mbit)
MBM29LV800TE (8Mbit)   S29AL016D (16Mbit)   HY29F200 (2Mbit)
MBM29LV800BA (8Mbit)   Am29LV010B (1Mbit)   HY29F400 (4Mbit)
MBM29LV800BE (8Mbit)   Am29LV002B (2Mbit)   HY29F800 (8Mbit)
MBM29LV160TE (16Mbit)   Am29LV200B (2Mbit)   HY29LV160 (16Mbit)
MBM29LV160BE (16Mbit)   Am29LV004B (4Mbit)   HY29LV162B (16Mbit)
MBM29DL161TE (16Mbit)   Am29LV040B (4Mbit)   HY29DL162T (16Mbit)
MBM29DL161BE (16Mbit)   Am29LV081B (8Mbit)   HY29DL163T (16Mbit)
MBM29DL162TE (16Mbit)   Am29LV008B (8Mbit)   HY29DL163B (16Mbit)
MBM29DL162BE (16Mbit)   Am29DS163D (16Mbit)   HY29LV320 (32Mbit)
MBM29DL163TE (16Mbit)   Am29N323D (32Mbit)   AT49BV8192 (8Mbit)
MBM29DL163BE (16Mbit)   Am29PDL128G (128Mbit)   AT49BV8192T (8Mbit)
MBM29DL164TE (16Mbit)   Am29PDS322D (32Mbit)   AT49LV8192 (8Mbit)
MBM29DL164BE (16Mbit)   LH28F400BG (4Mbit)   AT49LV8192T (8Mbit)
MBM29LV320TE (32Mbit)   LH28F800BG (8Mbit)   AT49BV160 (16Mbit)
MBM29LV320BE (32Mbit)   LH28F160BG (16Mbit)   AT49LV160 (16Mbit)
MBM29DL321TE (32Mbit)   LH28F400BV (4Mbit)   AT49BV160T (16Mbit)
MBM29DL321BE (32Mbit)   LH28F800BV (8Mbit)   AT49BV161 (16Mbit)
MBM29DL322TE (32Mbit)   LH28F160BV (16Mbit)   AT49LV161 (16Mbit)
MBM29DL322BE (32Mbit)   LH28F160BJ (16Mbit)   AT49BV161T (16Mbit)
MBM29DL323TE (32Mbit)   LH28F320BJ (32Mbit)   AT49LV161T (16Mbit)
MBM29DL323BE (32Mbit)   M28W160CT (16Mbit)   AT49BV320 (32Mbit)
MBM29DL324TE (32Mbit)   M28W160CB (16Mbit)   AT49BV320T (32Mbit)
MBM29DL324BE (32Mbit)   M28W320CT (32Mbit)   AT49BV321 (32Mbit)
MBM29DL32TF/BF (32Mbit)   M28W320CB (32Mbit)   AT49BV321T (32Mbit)
MBM29PL32TM/BM (32Mbit)   M28W640CT (64Mbit)   AT49LV320 (32Mbit)
MBMBM29DL640E (64Mbit)   M28W640CB (64Mbit)   AT49LV320T (32Mbit)
                              M29F400CT (4Mbit)   AT49LV321 (32Mbit)
                              M29F400CB (4Mbit)   AT49LV321T (32Mbit)
                              M29F800CT (8Mbit)   AT49BV640 (64Mbit)
                              M29F800CB (8Mbit)   AT49BV640T (64Mbit)
                              M29F160CT (16Mbit)
                              M29F160CB (16Mbit)

Verified SPI compatible flash devices:

Atmel AT25DF641 (64Mbits)   Cypress S70FL01GS (1Gbit)   Micron N25Q00A (1Gbit)
Atmel AT25DF321S (32Mbits)   Cypress S25FL512S (512Mbits)   Micron N25Q512 (512Mbits)
Atmel AT25DF321 (32Mbits)   Cypress S25FL256S (256Mbits)   Micron N25Q256 (256Mbits)
Atmel AT25DF161 (16Mbits)   Cypress S25FL256S (256Mbits)   Micron NP5Q128A (128Mbits)
Atmel AT25DF081 (8Mbits)   Cypress S25FL128S (128Mbits)   Micron N25Q128 (128Mbits)
Atmel AT25DF021 (2Mbits)   Cypress S25FL128S (128Mbits)   Micron N25Q064A (64Mbits)

Atmel AT26DF321 (32Mbits)	Cypress S25FL129P (128Mbits)	Micron N25Q064 (64Mbits)
Atmel AT26DF161 (16Mbits)	Cypress S25FL129P (128Mbits)	Micron N25Q032 (32Mbits)
Atmel AT26DF161A (16Mbits)	Cypress S25FL128P (128Mbits)	Micron N25Q016 (16Mbits)
Atmel AT26DF081A (8Mbits)	Cypress S25FL128P (128Mbits)	Micron N25Q008 (8Mbits)
Adesto AT45DB641E (64Mbits)	Cypress S70FL256P (256Mbits)	Micron M25P128 (128Mbits)
Adesto AT45DB642D (64Mbits)	Cypress S70FL256P (256Mbits)	Micron M25P64 (64Mbits)
Adesto AT45DB321E (32Mbits)	Cypress S25FL064 (64Mbits)	Micron M25P32 (32Mbits)
Adesto AT45DB321D (32Mbits)	Cypress S25FL032 (32Mbits)	Micron M25P16 (16Mbits)
Adesto AT45DB161E (16Mbits)	Cypress S25FL016A (16Mbits)	Micron M25P80 (8Mbits)
Adesto AT45DB161D (16Mbits)	Cypress S25FL008A (8Mbits)	Micron M25PX80 (8Mbits)
Adesto AT45DB081E (8Mbits)	Cypress S25FL040A (4Mbits)	Micron M25P40 (4Mbits)
Adesto AT45DB081D (8Mbits)	Cypress S25FL164K (64Mbits)	Micron M25P20 (2Mbits)
Adesto AT45DB041E (4Mbits)	Cypress S25FL132K (32Mbits)	Micron M25P10 (1Mbits)
Adesto AT45DB041D (4Mbits)	Cypress S25FL116K (16Mbits)	Micron M25P05 (512Kbits)
Adesto AT45DB021E (2Mbits)	Cypress S25FL216K (16Mbits)	Micron M25PX64 (64Mbits)
Adesto AT45DB021D (2Mbits)	Cypress S25FL208K (8Mbits)	Micron M25PX32 (32Mbits)
Adesto AT45DB011D (1Mbits)	Cypress S25FL204K (4Mbits)	Micron M25PX16 (16Mbits)
Adesto AT25SF161 (16Mbits)	Cypress S25FL004A (4Mbits)	Micron M25PE16 (16Mbits)
Adesto AT25SF081 (8Mbits)	MXIC MX25L51245G (512Mbits)	Micron M25PE80 (8Mbits)
Adesto AT25SF041 (4Mbits)	MXIC MX25L25655E (256Mbits)	Micron M25PE40 (4Mbits)
Adesto AT25XV041 (4Mbits)	MXIC MX25L256 (256Mbits)	Micron M25PE20 (2Mbits)
Adesto AT25XV021 (2Mbits)	MXIC MX25L12855E (128Mbits)	Micron M25PE10 (1Mbits)
Winbond W25M512 (512Mbits)	MXIC MX25L128 (128Mbits)	Micron M45PE16 (16Mbits)
Winbond W25Q256 (256Mbits)	MXIC MX25L6455E (64Mbits)	Micron M45PE80 (8Mbits)
Winbond W25Q128 (128Mbits)	MXIC MX25L640 (64Mbits)	Micron M45PE40 (4Mbits)
Winbond W25Q64 (64Mbits)	MXIC MX25L320 (32Mbits)	Micron M45PE20 (2Mbits)
Winbond W25Q32 (32Mbits)	MXIC MX25L3205D (32Mbits)	Micron M45PE10 (1Mbits)
Winbond W25Q16 (16Mbits)	MXIC MX25L323 (32Mbits)	Microchip SST25VF128B (128Mbits)
Winbond W25Q80 (8Mbits)	MXIC MX25L160 (16Mbits)	Microchip SST26VF064B (64Mbits)
Winbond W25Q80BW (8Mbits)	MXIC MX25L80 (8Mbits)	Microchip SST25VF064C (64Mbits)
Winbond W25Q40 (4Mbits)	MXIC MX25L40 (4Mbits)	Microchip SST26VF064 (64Mbits)
Winbond W25Q128FW (128Mbits)	MXIC MX25L20 (2Mbits)	Microchip SST25VF032B (32Mbits)
Winbond W25Q64FW (64Mbits)	MXIC MX25L10 (1Mbits)	Microchip SST25VF032 (32Mbits)
Winbond W25Q32FW (32Mbits)	MXIC MX25U643 (64Mbits)	Microchip SST26VF032 (32Mbits)
Winbond W25Q16FW (16Mbits)	MXIC MX25U323 (32Mbits)	Microchip SST26VF032B (32Mbits)
Winbond W25X64 (64Mbits)	MXIC MX25U163 (16Mbits)	Microchip SST26WF032 (32Mbits)
Winbond W25X64 (64Mbits)	MXIC MX25L512 (512Kbits)	Microchip SST26VF016 (16Mbits)
Winbond W25X32 (32Mbits)	MXIC MX25L1021E (1Mbits)	Microchip SST26VF032 (32Mbits)
Winbond W25X16 (16Mbits)	MXIC MX25L5121E (512Kbits)	Microchip SST26VF016B (16Mbits)
Winbond W25X80 (8Mbits)	Gigadevice GD25Q128 (128Mbits)	Microchip SST26VF016 (16Mbits)
Winbond W25X40 (4Mbits)	Gigadevice GD25Q64 (64Mbits)	Microchip SST26WF016B (16Mbits)
Winbond W25X20 (2Mbits)	Gigadevice GD25Q32 (32Mbits)	Microchip SST25VF016B (16Mbits)
Winbond W25X10 (2Mbits)	Gigadevice GD25Q16 (16Mbits)	Microchip SST26WF080B (8Mbits)
Winbond W25X05 (1Mbits)	Gigadevice GD25Q80 (8Mbits)	Microchip SST25VF080B (8Mbits)

EON EN25Q128 (128Mbits)　　Gigadevice GD25Q40 (4Mbits)　　Microchip SST25WF080B (8Mbits)
EON EN25Q64 (64Mbits)　　Gigadevice GD25Q20 (2Mbits)　　Microchip SST26WF040B (4Mbits)
EON EN25Q32 (32Mbits)　　Gigadevice GD25Q10 (1Mbits)　　Microchip SST25WF040B (4Mbits)
EON EN25QH32 (32Mbits)　　Gigadevice GD25Q512 (512Kbits)　　Microchip SST25VF080 (8Mbits)
EON EN25Q16 (16Mbits)　　Gigadevice GD25VQ16C (16Mbits)　　Microchip SST25VF040B (4Mbits)
EON EN25Q80 (8Mbits)　　Gigadevice GD25VQ80C (8Mbits)　　Microchip SST25WF040 (4Mbits)
EON EN25Q40 (4Mbits)　　Gigadevice GD25VQ41B (4Mbits)　　Microchip SST25WF020A (2Mbits)
EON EN25P64 (64Mbits)　　Gigadevice GD25VQ21B (2Mbits)　　Microchip SST25LF020A (2Mbits)
EON EN25P32 (32Mbits)　　Gigadevice GD25LQ128 (128Mbits)　　Microchip SST25WF020 (2Mbits)
EON EN25P16 (16Mbits)　　Gigadevice GD25LQ64 (64Mbits)　　Microchip SST25VF020 (2Mbits)
EON EN25F32 (32Mbits)　　Gigadevice GD25LQ32 (32Mbits)　　Microchip SST25WF010 (1Mbits)
EON EN25F16 (16Mbits)　　Gigadevice GD25LQ16 (16Mbits)　　Microchip SST25VF010 (1Mbits)
EON EN25F80 (8Mbits)　　Gigadevice GD25LQ80 (8Mbits)　　Microchip SST25WF512 (512Kbits)
EON EN25F40 (4Mbits)　　Gigadevice GD25LQ40 (4Mbits)　　Microchip SST25VF512 (512Kbits)
EON EN25F20 (2Mbits)　　Gigadevice GD25LQ20 (2Mbits)　　Microchip SST25VF020A (2Mbits)
ISSI IS25LD040 (4Mbits)　　Gigadevice GD25LQ10 (1Mbits)　　Microchip SST25VF010A (1Mbits)
ISSI IS25WQ040 (4Mbits)　　ESMT F25L04 (4Mbits)　　PMC PM25LV016B (16Mbits)
ISSI IS25LQ016 (16Mbits)　　ESMT F25L04 (4Mbits)　　PMC PM25LV080B (8Mbits)
ISSI IS25LQ080 (8Mbits)　　ESMT F25L08 (8Mbits)　　PMC PM25LV040 (4Mbits)
AMIC A25LQ64 (64Mbits)　　ESMT F25L08 (8Mbits)　　PMC PM25LV020 (2Mbits)
AMIC A25LQ32A (32Mbits)　　ESMT F25L32QA (32Mbits)　　PMC PM25LD020 (2Mbits)
AMIC A25L032 (32Mbits)　　Fidelix FM25Q16A (16Mbits)　　PMC PM25LV010 (1Mbits)
AMIC A25L016 (16Mbits)　　Fidelix FM25Q32A (32Mbits)　　PMC PM25LV512 (512Kbits)
AMIC A25LQ16 (16Mbits)　　Fidelix FM25M04A (4Mbits)　　Sanyo LE25FU406B (4Mbits)
AMIC A25L080 (8Mbits)　　Fidelix FM25M08A (8Mbits)　　Berg Micro BG25Q32A (32Mbits)
AMIC A25L040 (4Mbits)　　Fidelix FM25M16A (16Mbits)
AMIC A25L020 (2Mbits)　　Fidelix FM25M32A (32Mbits)
AMIC A25L010 (1Mbits)　　Fidelix FM25M64A (64Mbits)
AMIC A25L512 (512Kbits)　　Fidelix FM25M4AA (4Mbits)
AMIC A25LS512A (512Kbits)

Supported EEPROM devices that are SPI compatible

Atmel AT25010A (1Kbits)　　ST M95010 (1Kbits)　　ST M95640 (64Kbits)
Atmel AT25020A (2Kbits)　　ST M95020 (2Kbits)　　ST M95128 (128Kbits)
Atmel AT25040A (4Kbits)　　ST M95040 (4Kbits)　　ST M95256 (256Kbits)
Atmel AT25128B (128Kbits)　　ST M95080 (8Kbits)　　ST M95M01 (1Mbits)
Atmel AT25256B (256Kbits)　　ST M95160 (16Kbits)　　ST M95M02 (2Mbits)
Atmel AT25512 (512Kbits)　　ST M95320 (32Kbits)　　Microchip 25AA160A (16Kbits)
Microchip 25AA160B (16Kbits)

Supported MCU's with on board memory programmable via SPI

Nordic nRF24LE1　　Altera EPCS1　　Altera EPCS4
Nordic nRF24LU1+　　Altera EPCS16　　Altera EPCS64
Altera EPCS128

**Supported MCU's with on board memory programmable via JTAG**

Xilinx XC2C32A	Xilinx XC2C64A	Xilinx XC2C128
Xilinx XC2C256	Xilinx XC2C384	Xilinx XC2C512
Xilinx XC9500XL	Xilinx XC95288XL	Xilinx XC95144XL
Xilinx XC9572XL	Xilinx XC9536XL	Lattice LC4032V
Lattice LC4064V	Lattice LCMXO256	Lattice LCMXO640
Lattice LCMXO1200	Lattice LCMXO2280	

If your flash is not listed above, contact us via email at: contact@embeddedcomputers.net. 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, depending on what operating system you are using, you may have to manually install the driver. To do this:

1. Open the run command by pressing and holding the Windows key, then press the R key ("Run").

2. Type devmgmt.msc

3. The Device Manager window will now appear. Find the newly connected device, usually in the "Other devices" category.

4. Right-click the Unknown device and select "Update driver software".

5. You will be presented with two choices, "Search automatically..." and "Browse me computer...". Select the later to browse for the driver.

6. Select the folder "RC18 Release\Drivers". You can use this driver for all FlashcatUSB firmware, including SPI, JTAG, NAND, and DFU.
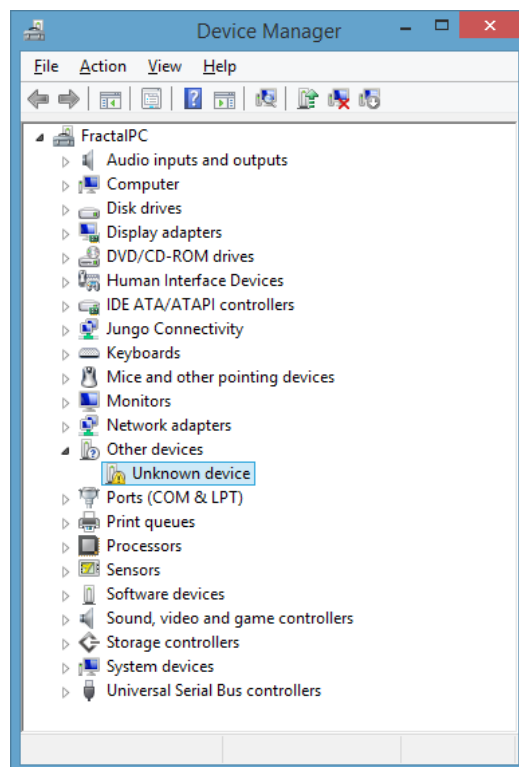
Because a different USB device is created with each AVR firmware, you will need to repeat this process once for each AVR firmware you install into FlashcatUSB.

HARDWARE LAYOUT
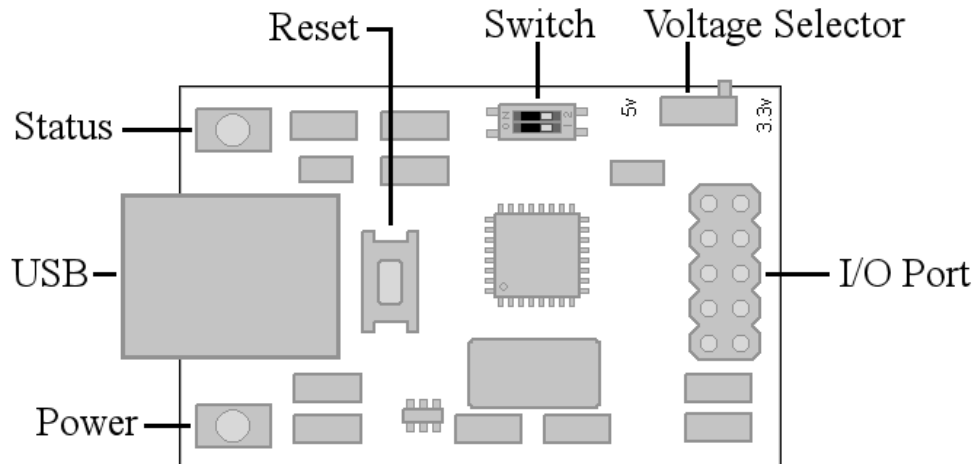
Definitions of diagram labels:

Status – This LED (blue) blinks when the device is reading or writing to memory. This LED will be solid to indicate that the board is successfully connected to the FlashcatUSB interfacing software.

Power – This LED (red) turns solid when the device is powered on and the on-board firmware application has started.

USB – This is the USB Type-B that needs to be connected to a USB cable that is then connected to your PC.

Reset – This button will reset the firmware application. If the switch #2 has been set to off, the board will instead boot into DFU mode.

Switch – This switch has two pins, the first is application specific and is intended to be used in future software updates. The second pin is used to set the device in application mode or DFU mode.

Voltage Selector – This switch controls the voltage output. If the switch is at the left position, the board will operate and output at 5v. If the switch is at the right position, the board will operate and output 3.3v.

I/O Port – This is the 10-pin port that is connected to your target device or Flash memory.

USING BOOTLOADER MODE TO CHANGE AVR FIRMWARE

FlashcatUSB is shipped with the latest SPI firmware installed. This means that out of the packaging, you can use the device to read and write to any SPI device. However, if you wish to use a different protocol, such as JTAG or NAND mode, then you must change the firmware on the device itself. You can change the device's firmware over USB using the built-in bootloader mode.

When in bootloader mode (also known as DFU mode), the FlashcatUSB software will allow you to reprogram the device using any Atmel AVR compatible HEX file. In this software package, we include SPI firmware, JTAG firmware, and NAND firmware files located in the software "Firmware" folder.

To put the device into Bootloader Mode, simply set switch #2 from ON to the OFF position. As shown in the diagram below:
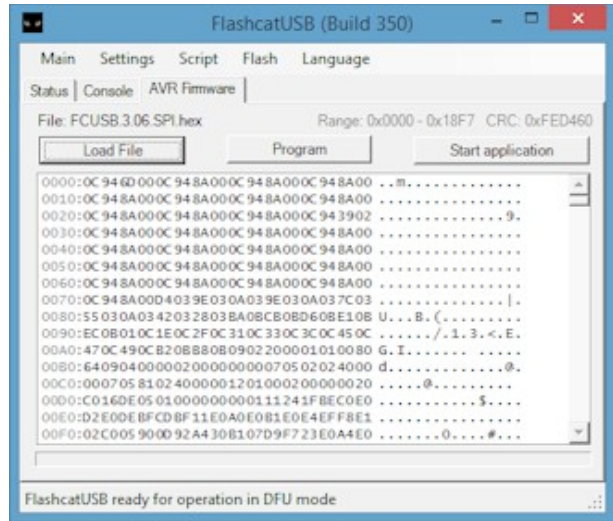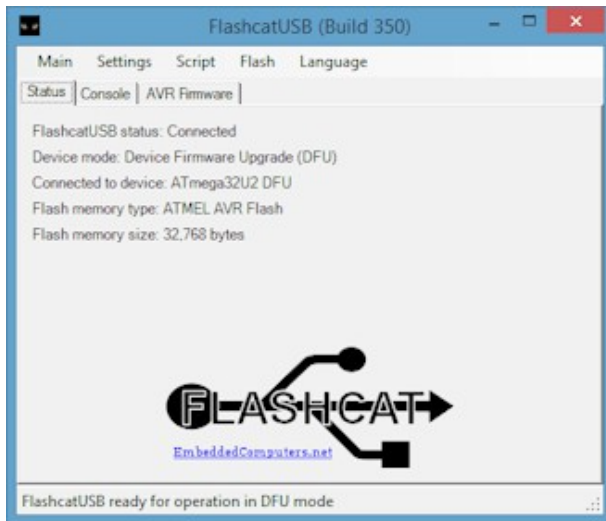

*Application Mode*   *Bootloader Mode*

With the hardware in Bootloader mode, each time the device is reset (by pressing the RESET button), FlashcatUSB will startup in bootloader mode. With the software running, the main screen will now show that the device is in bootloader mode ("Device Firmware Upgrade") and that it is ready to be reprogrammed with AVR firmware.

To change AVR firmware, click 'Load File', select the firmware HEX file, click 'Program', and once complete, click the 'Start Application' button. After you have successfully changed firmware, we recommend that you set switch #2 to ON, to prevent the device from booting back into DFU mode.
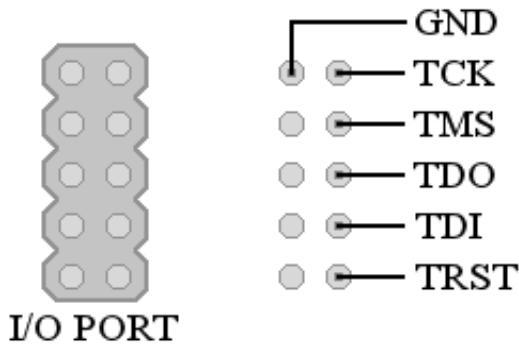
USING JTAG MODE FOR FLASH PROGRAMMING

With the "JTAG" 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.
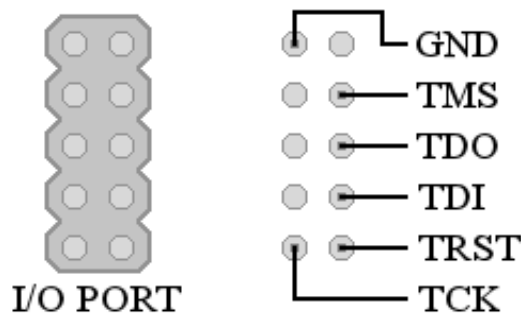
PCB 2.x pinouts for JTAG

The image above shows you the pin outs of the 10-pin I/O 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. Note: the TDO pin on FlashcatUSB connects to the TDO pin on the target JTAG port and same for TDI, as they are not swapped like the SPI connection labels.

Please note that the pinouts for JTAG has been changed in PCB version 2.x. This is because the top-right most pin is now a dedicated VCC line and thus can not be used for a general I/O pin like in previous versions. If you have an older PCB version, the legacy pinout is:


PCB 1.x legacy pinouts for JTAG

USING SPI MODE FOR FLASH PROGRAMMING

With the SPI mode firmware installed, you can use the device as a high-speed programmer for virtually every SPI and SQI compatible Flash memory device.
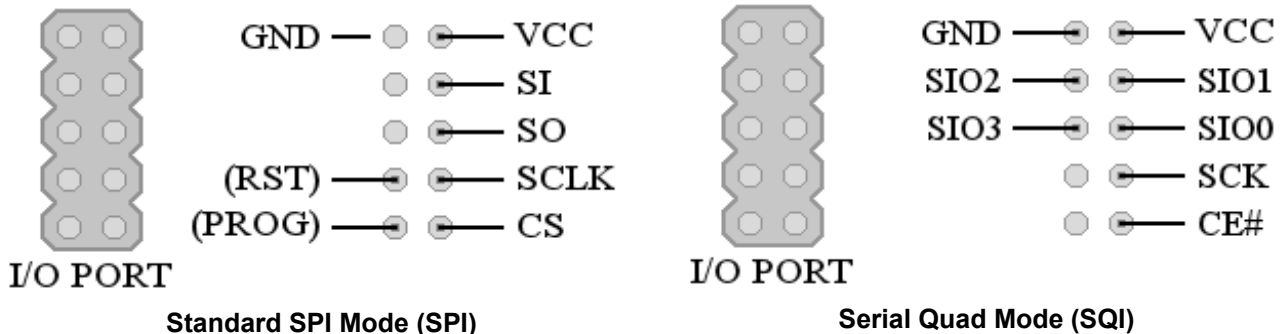
**Standard SPI Mode (SPI)** **Serial Quad Mode (SQI)**

The image above shows you the pin outs of the 10-pin I/O port and how it should be connected to the SPI or SQI bus of your targeted 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. For devices that operate with 1.8v, Embedded Computers offers a low-cost 3.3v to 1.8v adapter. For SPI connections, the SI on FCUSB connects to the SO on the Flash memory. For SQI, connect SIO0 to SIO0 and so on.
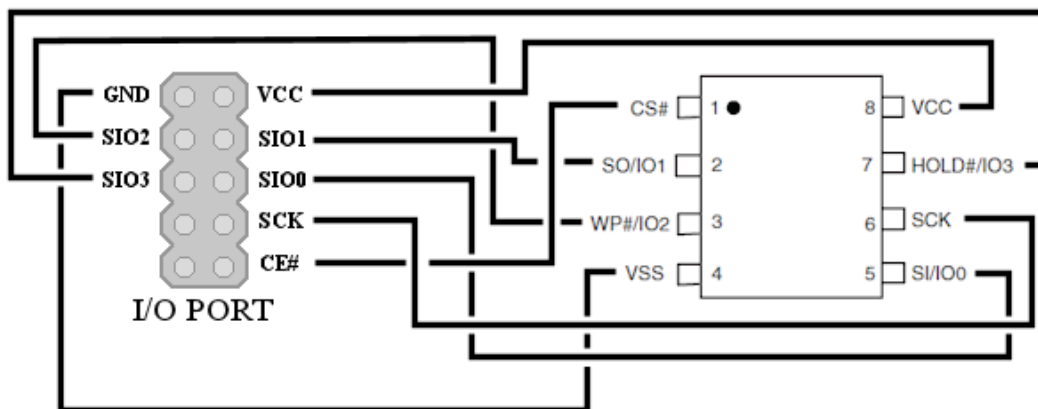
***Note**: the (RST) and (PROG) pins are only used when connecting to MCU's with on board flash, such as the Nordic nRF24LE1 device.
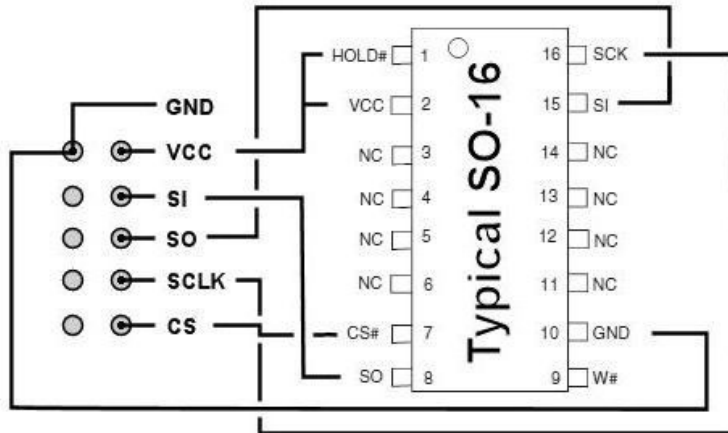
The above diagram is how you should connect the 10-pin port to a typical SOIC-16 package of a SPI compatible Flash device.
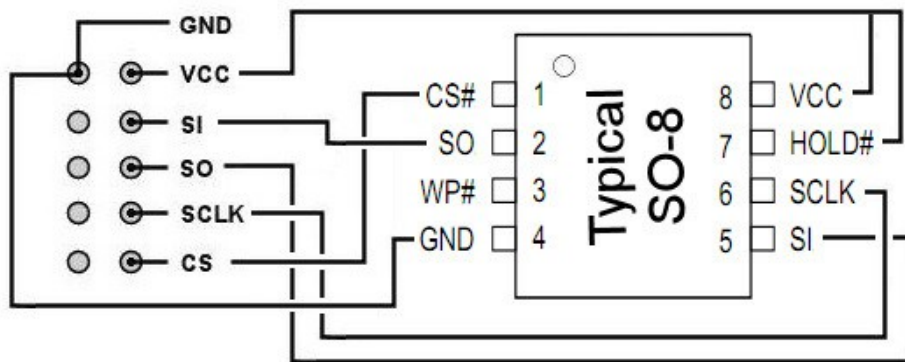
The above diagram is how you should connect the 10-pin port to a typical SOIC-8 and DIP-8 package of a SPI compatible Flash device.

This diagram shows how to connect FlashcatUSB to a Quad I/O (SQI) capable Flash device (SO8 package). The software will automatically detect that a device is connected in this manner and will automatically send the commands needed to enter quad-mode.
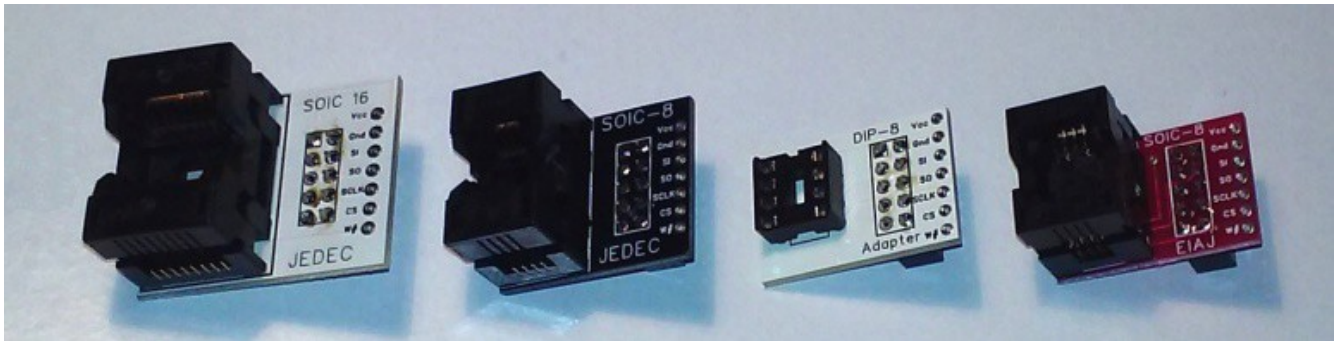
Optionally, you can also purchase drop-in sockets and in-circuit clips for most common SPI chip packages. Embedded Computers carries many of these in stock.
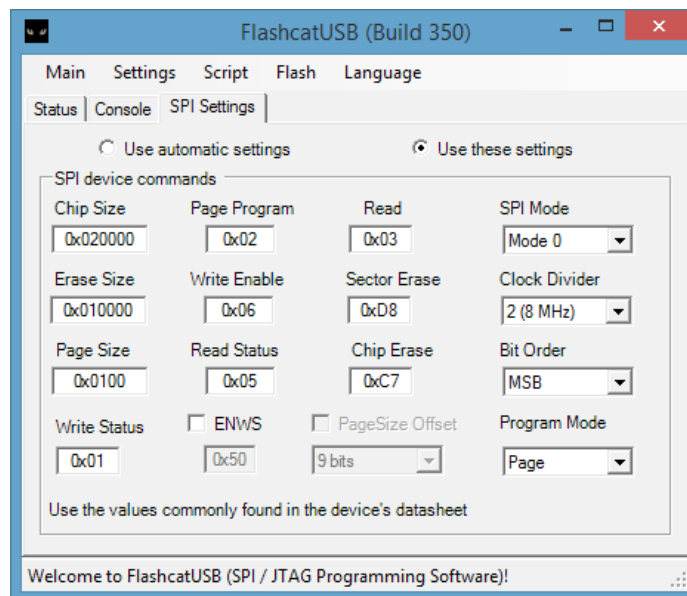
When you start FlashcatUSB with the SPI device connected, the software should automatically detect the chip and load its settings accordingly. If the chip is detected, but not supported by the software, you can then use the SPI Settings tab to manually specify all of the settings with the values from 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
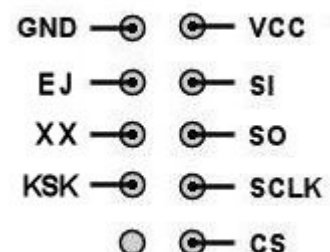
SPI) and you can also use FlashcatUSB to program those types of chips.

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

USING SPI MODE TO PROGRAM MSI MOTHERBOARDS

FlashcatUSB can be used to program the BIOS memory of many different motherboards from a variety of manufacturers. Some may require a SOIC clip, while others, such as MSI, have a on-board pin header.

First locate the IDC pin-header on the motherboard labeled JSPI1. Notice, that this pin-header is often omitted from the manual, so you will need to find it physically on the board.
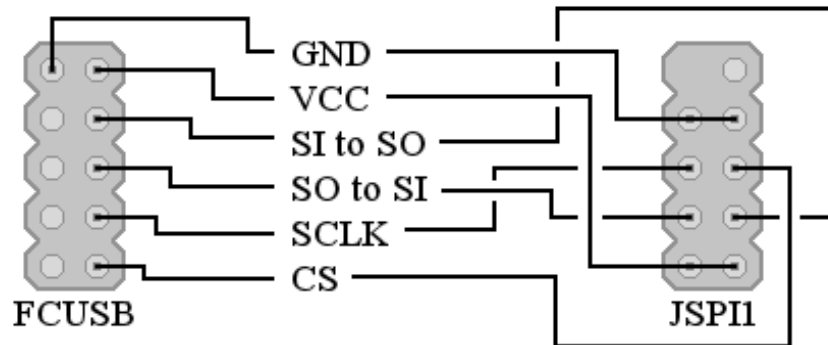
Notice that the GND is connected to 2 pins on JSPI1, pin 7 and pin 8. VCC is connected to pin 1 and pin 2. Pin 9 (the top left pin) is not connected to anything.

NOTE: some motherboards do not have an isolated VCC circuit, in which case, FlashcatUSB will automatically disconnect when trying to self-power the board. In this circumstance, you will need to power the motherboard normally using the PSU, and do not connect the VCC line from FlashcatUSB. This will allow the chip to be powered by the on board circuit instead.

USING THE SPI MODE TO PROGRAM I2C EEPROM
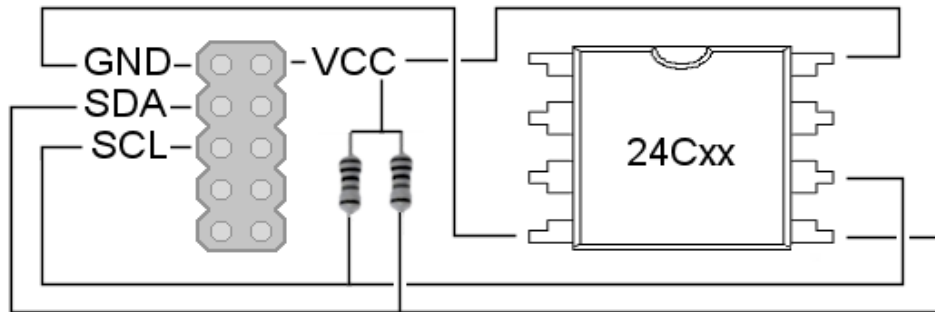
To program a I2C or TWI compatible EEPROM device, FlashcatUSB must be loaded with AVR firmware 4.01 or newer. Notice, this feature is only available on PCB 2.x and will not work with older PCB 1.x boards.
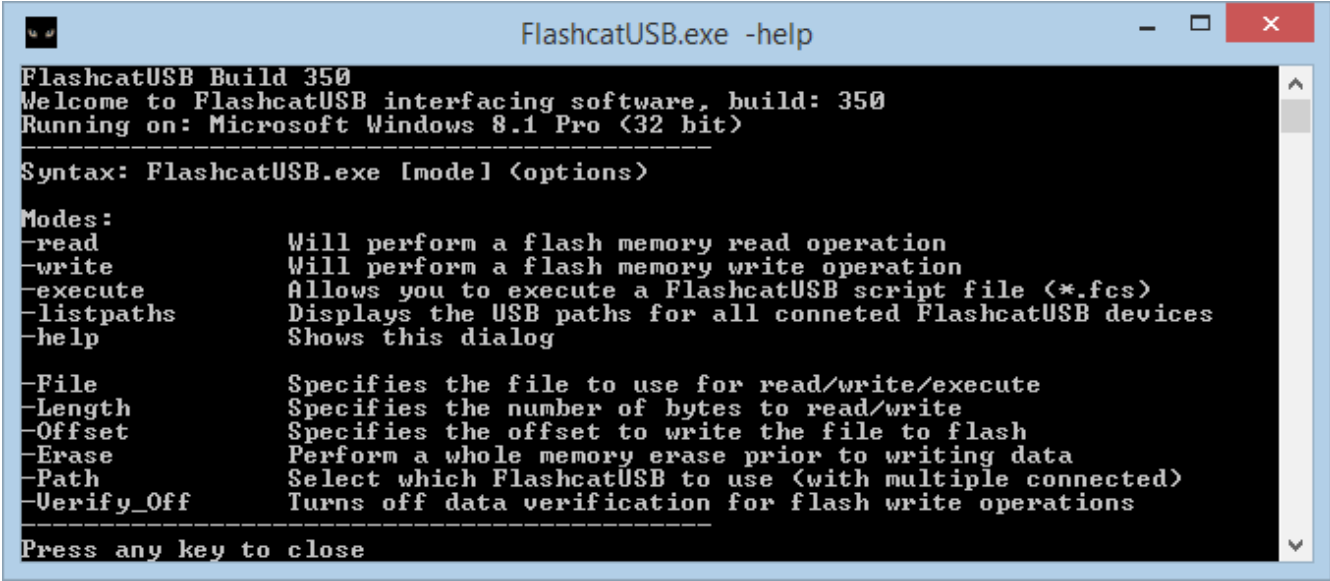
The wiring connection only uses 4 pins from the FlashcatUSB I/O port. When connecting to an EEPROM device, the SDA and SCL lines must also have a resistor that connects to VCC. We recommend using 4.7K values, although any resistor from 4K to 10K should be suitable. If you are using the SPI DIP (revision 2) adapter, you do not need any resistors, as the adapter already contains them.
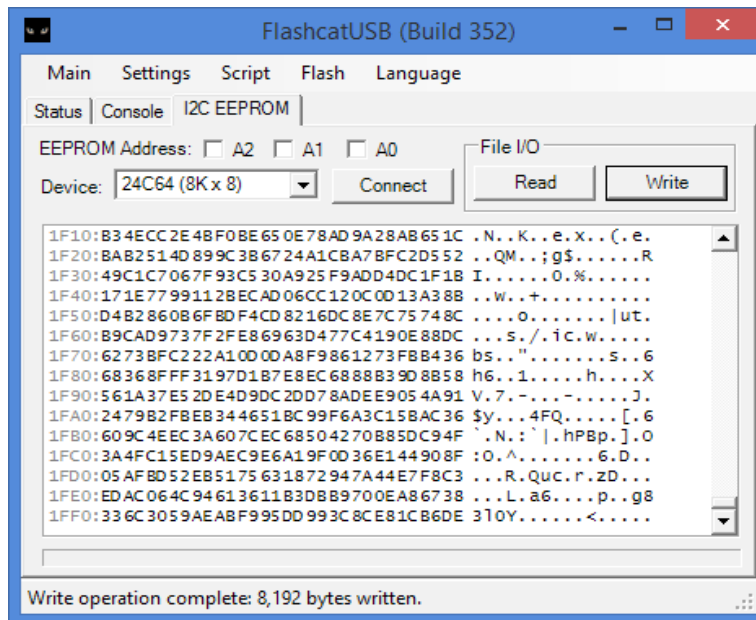
With the device successfully connected, run the software and from Setting menu, select 'I2C EEPROM Mode'. Then click 'Detect' from the 'Main' menu. A new tab will appear that will allow you to read and write to the EEPROM.

USING THE COMMAND PROMPT / CONSOLE MODE

FlashcatUSB can also be operated from the Window's command prompt. This will allow you to create batch files that are useful for production environments. To get started, open a command prompt window, go to the folder where FlashcatUSB.exe is, and type:

FlashcatUSB.exe -help

This will display a list of commands you can use.

To use this feature, call FlashcatUSB.exe and specify which mode you want to do. This is limited to reading, writing data, or executing a script file. Additional parameters are then added to specify certain settings. For example:
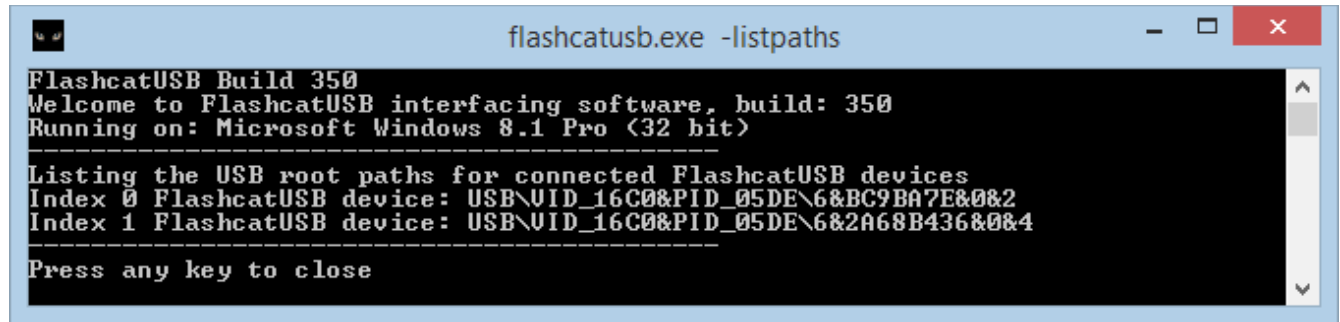
FlashcatUSB.exe -read -file flash.bin

Will read the entire data from the flash memory and save it to flash.bin (in the same directory FlashcatUSB.exe is in).

Multiple FlashcatUSB on a single machine can also be used. First run the command:

FlashcatUSB.exe -listpaths

This will display a window similar to this:


```
flashcatusb.exe  -listpaths                              _  □  ×

FlashcatUSB Build 350
Welcome to FlashcatUSB interfacing software, build: 350
Running on: Microsoft Windows 8.1 Pro (32 bit)
--------------------------------------------------
Listing the USB root paths for connected FlashcatUSB devices
Index 0 FlashcatUSB device: USB\VID_16C0&PID_05DE\6&BC9BA7E&0&2
Index 1 FlashcatUSB device: USB\VID_16C0&PID_05DE\6&2A68B436&0&4
--------------------------------------------------
Press any key to close
```

In this example, a FlashcatUSB device is connected to
USB\VID_16C0&PID_05DE\6&2A68B436&0&4

Now you can specify that in the command line using the -PATH parameter, such as:

FlashcatUSB.exe -read -file out.bin -path "USB\VID_16C0&PID_05DE\6&2A68B436&0&4"

Note: use quotes to encapsulate the path string.

GETTING STARTED WITH THE SCRIPTING ENGINE

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, events, and condition/loops. When a script file is ran, 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

Variables

Events / Functions

Conditional Statements (IF … ELSE … ENDIF)

Loops / Iterations (FOR … ENDFOR)

AUTORUN Feature

Basic Syntax

FlashcatUSB scripts use a very familiar, VB-style, 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 or 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.

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.

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.

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 msgbox 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 msgbox 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. 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.

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.

```
For (i = 0 to 9)
    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.

      File 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)

File I/O Commands

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:	ReadFile
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 = ReadFile("Scripts\EEPROM.bin")

Command:	WriteFile
Parameters:	Data, String
Description:	Writes an array of data to the hard drive. The first parameter is the data array variable, the second is the location in relation to FlashcatUSB.exe where you want to save the file. This command does not prompt the user.
Examples:	WriteFile(MyData,"Scripts\EEPROM.bin")

Memory Commands

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)

GUI Commands

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:	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")

JTAG Specific Commands

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:	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:	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.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

SPI Specific Commands

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:	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:	Integer or Data
Description:	Writes data to the status register.
Examples:	SPI.SetSR(0xF0)

Miscellaneous Commands

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

Command:	Verify
Parameters:	Optional 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:	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:	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:	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