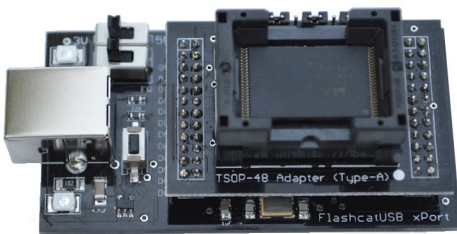
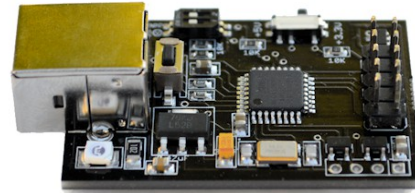


# *Embedded Computers*

For professionals



## FlashcatUSB

### USER'S GUIDE

### Classic, Professional, xPORT, and Mach<sup>1</sup>

Website: [www.embeddedcomputers.net/products/FlashcatUSB/](http://www.embeddedcomputers.net/products/FlashcatUSB/)  
Support email: [contact@embeddedcomputers.net](mailto:contact@embeddedcomputers.net)  
Last updated: February, 2019



# INDEX

[Introduction](#)  
[Memory hardware support](#)  
[Software requirements](#)  
[Driver installation](#)  
[Hardware layout](#)  
[Software overview](#)  
[Bootloader mode](#)  
[Multi-device programming](#)  
[JTAG mode for Flash programming](#)  
[SPI mode for Flash programming](#)  
[SPI mode to program motherboards](#)  
[How to program I2C EEPROM](#)  
[How to program Microwire EEPROM](#)  
[How to program One-wire EEPROM](#)  
[Software ECC for NAND devices](#)  
[JTAG Boundary Scan Programming](#)  
[Using the command prompt / console mode](#)  
[List of supported Flash memory devices](#)

## WHATS NEW

1. Professional can now do JTAG boundary scan programming (CFI NOR)
2. FlashcatUSB Mach<sup>1</sup> now has ability to program HyperFlash
3. FlashcatUSB Classic can now program one-wire EEPROM
4. Console mode has been upgraded
5. FlashcatUSB Professional now supports JTAG operation
6. Added German language option to software
7. Hex Editor control now allows for real-time editing mode
8. FlashcatUSB Professional now supports QUAD SPI mode

# INTRODUCTION

FlashcatUSB is a versatile multi-protocol Flash memory programming platform. This family of products includes: **Classic**, a low-cost 8-bit microcontroller based design, **Professional**, a higher performance 32-bit ARM based microcontroller with ISP logic, and **xPORT**, a programmer designed to interface with NOR/NAND memory in high-pin packages. **Mach<sup>1</sup>** is the most advanced programmer with high-speed USB and a high performance logic device to interface with the highest density memories available.

FlashcatUSB is a cost effective hardware tool that is used both by industry professionals, semiconductor manufacturers, defense contractors, and hobbyists. By utilizing a USB microcontroller 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.

## **Classic hardware features:**

ATMEGA32U2 8-bit AVR microcontroller  
Processor specifications: 16MHz clock, 8Mhz peripheral clock  
USB PHY: Full-speed mode (12Mbps)  
Memory: 1KB RAM, 32KB Flash (4KB reserved for bootloader)  
Voltage output: 3.3v or 5.0v (current @ 1000ma)  
Reverse voltage protection (STMP52171 MOSFET)  
VCC output disable switch

## **Professional hardware features:**

SAM3U 32-bit ARM microcontroller  
Processor specifications: 96MHz, 48MHz peripheral clock  
USB PHY: High-speed mode (480Mbps)  
Memory: 36KB RAM, 128KB Flash (16KB reserved for bootloader)  
Voltage output: 1.8v and 3.3v (current @ 500ma)  
On board CPLD (Altera 5M160Z) with 160 programmable logic elements  
VCC output disable switch

## **xPORT hardware features:**

ATMEL AT90USB646 8-bit AVR microcontroller  
USB PHY: Full-speed mode (12Mbps)  
Processor specifications: 16MHz clock, 8Mhz peripheral clock  
Memory: 4KB RAM, 128KB Flash (8KB reserved for bootloader)  
Voltage output: 3.3v or 5.0v (current @ 300ma)

## **Mach<sup>1</sup> hardware features:**

ATMEL SAM3U 32-bit ARM microcontroller  
Processor specifications: 96MHz, 48MHz peripheral clock  
USB PHY: High-speed mode (480Mbps)  
Memory: 52KB RAM, 256KB Flash (16KB reserved for bootloader)  
Voltage output: 1.8v and 3.3v (current @ 500ma)  
On board CPLD (Altera 5M570Z) with 570 programmable logic elements

**SPI Mode supported features:**

Mode 0, 1, and 2 compatible  
SPI-QUAD and SPI-DUAL IO supported  
High density devices supported: 1 to 128 mbit. (24-bit addressing)  
Ultra-high density devices supported: 256 mbit to 2 Gbit (32-bit addressing)  
Ability to program MCU's with on board Flash / NV memory  
SPI-NAND devices are supported (up to 4Gbit)

**I2C mode supported features:**

Supports all I2C and TWI memory devices  
Address selectable (A0, A1, A2)  
Supports up to 400kHz data rates (1MHz for Pro)

**JTAG mode (PCB 2.x) supported features:**

Clock speed: 1MHz  
CFI compatible flash memory – Intel, AMD, and SST algorithms supported  
DMA mode supported for target memory access  
MIPS supported (for EJTAG)  
Instruction register (IR) auto-selected from 4 to 512 bits.

**JTAG mode (PCB 4.x) supported features:**

Clock speed: 40MHz, 20MHz and 10MHz selectable  
Universal JTAG support (MIPS, ARM, Toshiba, Intel, etc.)  
Boundary Scan programming for parallel Flash devices.

**Parallel Flash mode supported features:**

Extension Port: Parallel NOR devices (up to 512Mbit)  
FlashcatUSB xPORT: Parallel NOR devices (up to 2Gbit)  
NAND support: X8 IO, 3.3V, SDR  
OTP EPROM (27 series) with +12V VPP support

**Mach<sup>1</sup> mode supported features:**

NAND support: X8/X16 IO, 1.8V/3.3V, SDR/DDR  
HyperFlash S26KL/S26KS, 1.8V/3.3V

**Software ECC algorithms supported:**

Hamming (1-bit correction)  
Reed Solomon (1-bit to 14-bit correction)  
Binary BHC (1-bit to 14-bit correction)

## MEMORY HARDWARE SUPPORT

Below is a list of all the types of memory devices supported and which programmers can be used.

	<b>Classic</b>	<b>XPORT</b>	<b>Professional</b>	<b>Mach<sup>1</sup></b>
SPI	X	X (Note 1)	X	X
SPI NAND	X (Note 2)	X (Note 2)	X	X
SPI QUAD IO	X (Note 8)	X (Note 8)	X (Note 7)	X
I2C	X	X	X	
Microwire	X	X	X	
1-Wire	X		(Note 6)	
X8 NAND	X (Note 3)	X (Note 5)	X (Note 3)	X
X16 NAND				X
HyperFlash				X
Parallel Flash	X (Note 4)	X	X (Note 4)	(Note 6)
Voltage: 5V	X	X		
Voltage: 3.3V	X	X	X	X
Voltage: 1.8V			X	X

### Notes:

- 1: SPI adapter required
- 2: Speed is limited to software implementation
- 3: Extension port adapter required
- 4: Devices up to 512Mbit in X16 supported
- 5: SLC devices up to 4Gbit supported
- 6: Planned for future update
- 7: Supports single, dual and quad operations
- 8: Speed limited to 200KB/s read and 100KB/s write

# SOFTWARE REQUIREMENTS

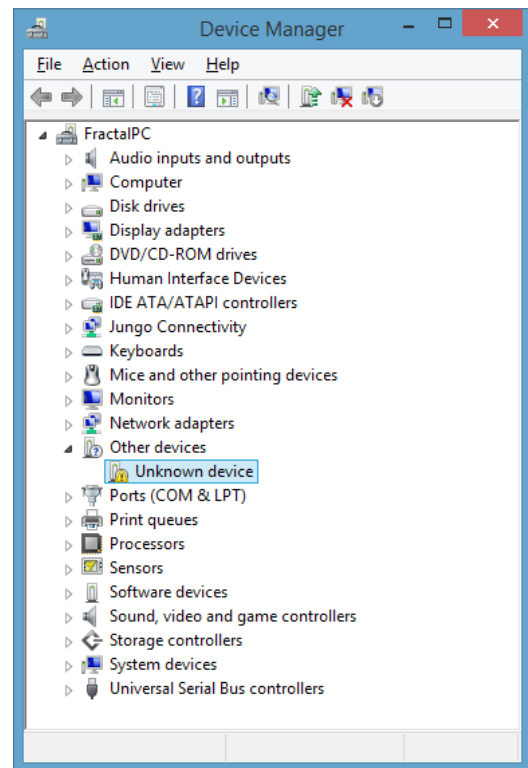
A computer with at least a 1 GHz processor and 256 MB of free memory available, and a USB 1.1 / 2.0 port. Operating systems supported: Windows XP, Windows Vista, Windows 7, 8, 8.1, and 10. Supports both 32-bit and 64-bit versions.

Microsoft .NET Framework 4.0 ([Download](#))

# DRIVER INSTALLATION

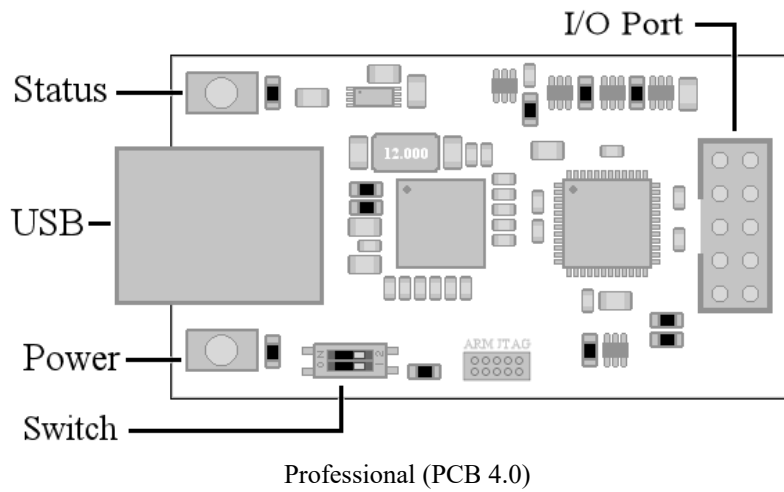
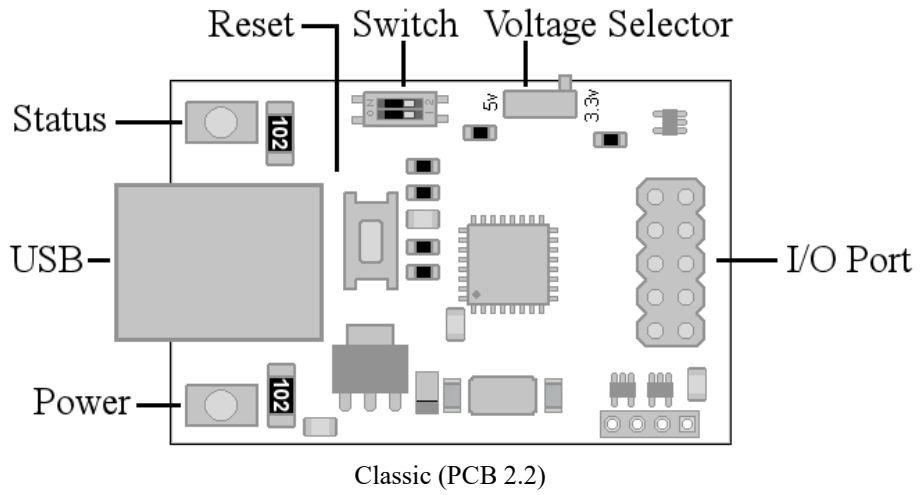
When you connect FlashcatUSB to your PC for the first time, you should notice a "Found new hardware" notification pop-up, depending on what operating system you are using, you may have to manually install the driver. To do this:

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 "Drivers". You can use this driver for all FlashcatUSB firmware, including SPI, JTAG, NAND, and DFU.



You may need to do this for each device mode. You should use the same driver file for every FlashcatUSB device you connect to your PC.

# 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 part has two switches, the first enables/disables the VCC out pin. The second pin is used to enable bootloader mode, if this pin is off, pressing the RESET button will cause the device to reset and start in bootloader mode. This mode is used to program the main application firmware.

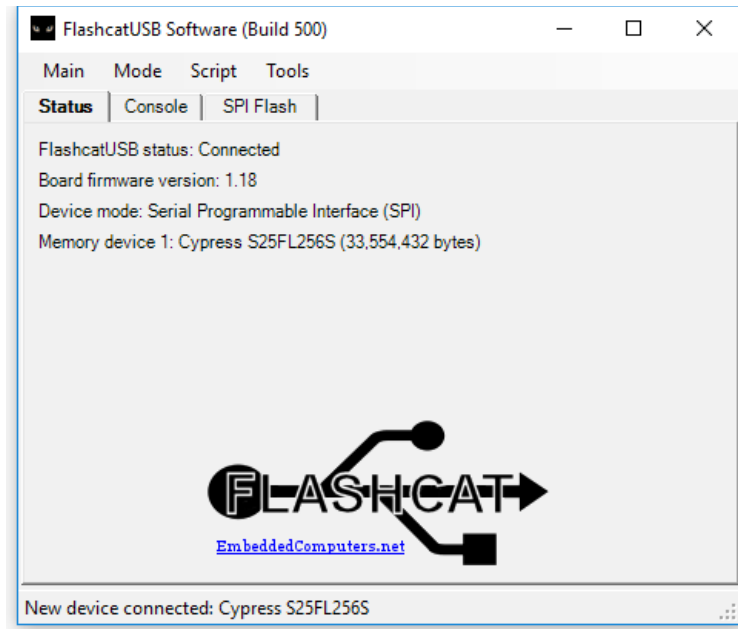
*On newer revision boards, a label of **APP** and (**V**) are used indicate which pins are used for the application/bootloader mode and which pin is used to disable the VCC output pin.*

**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. The top left pin is a dedicated ground connection, the top right pin is the dedicated voltage supply.



# SOFTWARE OVERVIEW



The open-source software will run on Microsoft Windows operating system from Windows XP to newer. If you already connect FlashcatUSB to your PC, when you run the software it will automatically attempt to connect to any attached Flash memory (using the operation mode last used). Otherwise, connect your device, set the operation mode and then click the Detect Device menu option.

## Main Menu

**Detect Device** – will initialize the software and attempt to connect to a Flash using the current settings and operation mode.

**Repeat write operation** – will automatically perform the last Flash write operation. This feature is useful for programming multiple Flash devices using the same data.

**Refresh flash device** – will cause the current Flash tab and hex editor to refresh its data screen (performs a small read operation from the Flash device).

**Exit** – will close down the software.

## Settings Menu

**Protocol Settings** – this will bring up the window to set protocol specific settings, such as SPI clock frequency, i2c address, and NAND memory settings / bad block manager / layout.

**Verify Programming** – this setting will make all write operations verify the data written after each block/sector write by reading back the data and comparing it. If the comparison fails, the data is then re-written up to 3 times.

**Bit Swapping** – is a tool that will automatically swap the bits from data operations. The software supports swapping every 4-bits, 8-bits, and 16 bits. For example, 0x1F with 4-bit swapping would appear as 0xF1.

**Endian Mode** – is also a bit modification tool that will change the base endian (byte order) of the data. That is the MSB and LSB. For example, 0x31 with Little Endian mode will output 0x8C. This tool can be used if the Flash memory is used as a microprocessor boot device that executes code in a specific byte endian.

## Script Menu

**Current script** – in JTAG mode, this will display all of the compatible scripts that are specified for your processor. You can also use it to change/select other compatible scripts.

**Load script** – this allows you to manually load a specified script.

**Unload script** – this will unload any current or loaded script.

## Tools Menu

**Erase chip** – will perform a full chip erase of the selected Flash device. Please note, not all Flash devices support full chip erase, in which case, the software will erase all sectors.

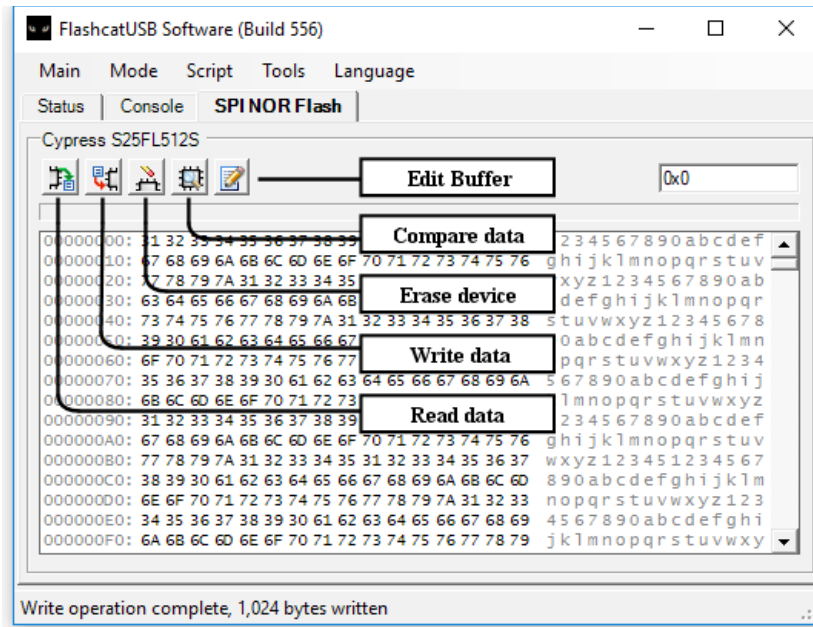
**Create image** – this option will do two full chip reads, compare the data to ensure a good read, and create a single zip file with entire data compressed.

**Write image** – this tool will open an image created using the Create image tool and write the data into a new Flash device.

**NAND memory map** – will launch the NAND memory map tool, which will show a graphical grid of all the NAND blocks. You can also use this tool to perform a full block erase and random byte read to verify if the block is still good.

**Vendor Features** – some devices may have vendor specific menus. This menu item will bring up that menu to access / change these settings. Some examples include SPI devices that have non-vol registers for changing SPI to QUAD mode. In single-wire mode, this feature will allow you to read/write to the security and OTP memory area.

# The Flash Tab



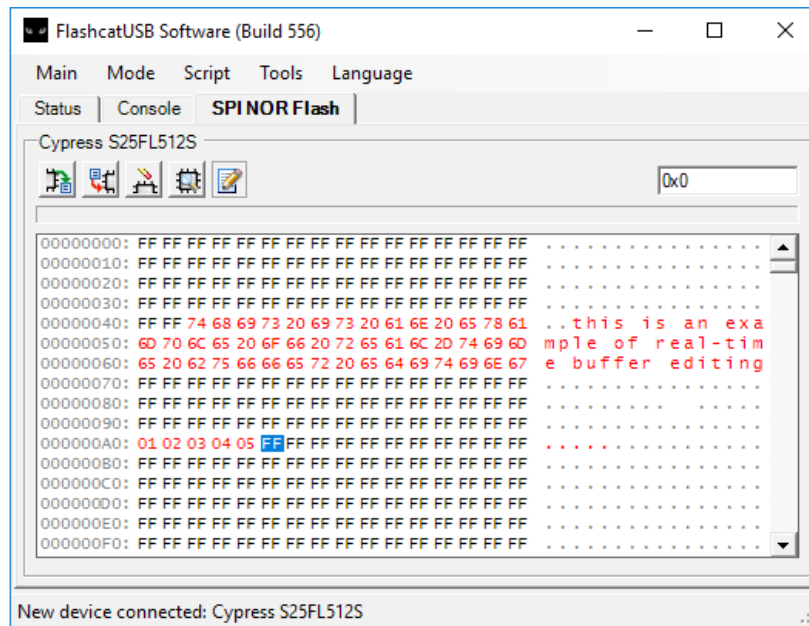
With a Flash device successfully detected, the software will create a Flash tab. This is the primary tool you will use to interact with a Flash device. This window will show you the name of the Flash device, as well as a hex data grid (and ASCII window) representation of the current data on the device. You can scroll down to see the data update in real-time. (Tip: pressing F5 will refresh the screen)

There are five buttons: read data, write data, erase all, compare, and edit mode. When you click the read data button, a window will appear to allow you to specify a base address and length. By default, the base address will be zero and the length will be the size of the entire Flash, so that if you do not change anything, the default operation will be to read the entire chip starting from the first byte. After clicking OK, a prompt will ask you where to save the data and it will allow you to save in three file formats: binary (\*.bin), Intel HEX (\*.hex) and Motorola S-Records (\*.srec). This can be any location on your computer.

If you click the write data button, a file open prompt will appear asking for a file. By default it will filter for only binary files (\*.bin), but you can use the drop down menu to also select Intel HEX files (\*.hex) or Motorola S-Records (\*.srec) which will automatically convert those file formats into binary data for writing. Upon selecting a file, you will now be able to set the base address and length of data to write, the default choice is the beginning of Flash and the length of the file or the Flash device, whichever is shorter.

The compare button will allow you to read a file from disk and compare it to the contents on the Flash. This will result in a window telling you how many bytes matched and a percentage.

The Edit Mode will allow you to edit the data buffer and write it to the Flash, while automatically filling the buffer with existing data. To use this feature, click the Edit Mode button and then move your cursor to any area in the hex editor or ASCII area and click.



If you edit the hex area, you can automatically change the next byte by pressing the Enter key.

In the ASCII area, you can type a single or multiple characters that will automatically be converted to hex using the standard ASCII to hexadecimal conversion table.

All changed data will be represented with red characters.

Finally, when you are satisfied with the edits, click the Edit Mode button once again which will prompt you if you want to write changes. Select Yes and the software will automatically update all sectors with changed data.

## BOOTLOADER MODE

FlashcatUSB Classic 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 SWI, or you need to update the firmware, 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.

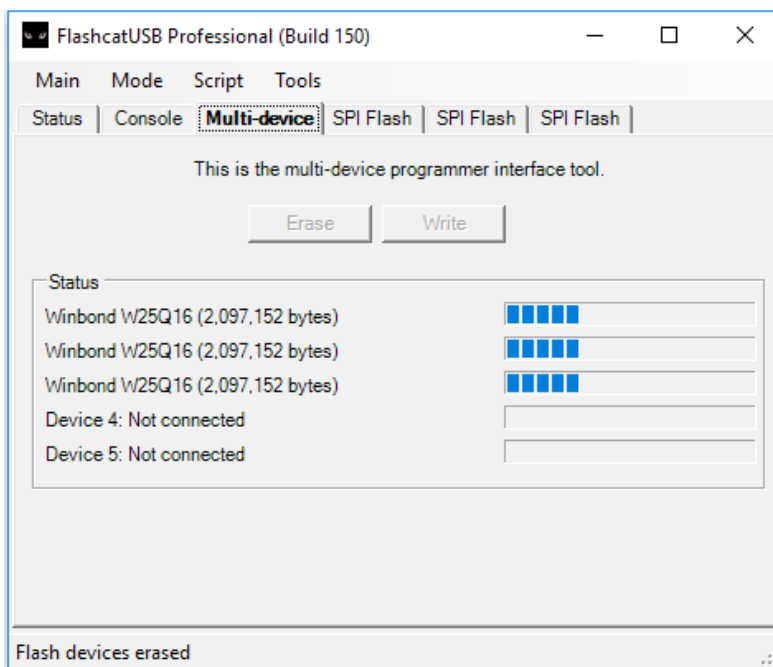
To change AVR firmware, click 'Load File', select the firmware HEX file, click 'Program', and once complete, click the 'Start Application' button. After you have successfully changed firmware, we recommend that you set switch #2 to ON, to prevent the device from booting back into DFU mode.

FlashcatUSB xPORT has a single toggle switch with the label "FW" below it. Set the toggle switch to the right and press the button to change to bootloader mode. Please note that the xPORT does not use the same firmware file as Classic.

Note: FlashcatUSB Professional has automatic firmware updating via software. So when the software first connects to the Professional board via USB, it will check the firmware version and if needed will perform a unit update.

## MULTI-DEVICE PROGRAMMING

Up to five FlashcatUSB devices can be connected to a single PC to allow for multi-device programming simultaneously (also known as gang programming).



When two or more FlashcatUSB are connected, the software will display the “Multi-device” tab as shown above. From this tab, you can erase or write to all detected memory devices at once. This is ideal for production environments that need to program hundreds, or even thousands of memory devices daily.

When using this mode, its best to connect the FlashcatUSB to native USB ports on your PC. This is because each device uses the maximum allowed current available in the USB specification (5v @ 500ma). So if you try and use a USB hub, it may not be able to provide enough current to power multiple devices, although it is safe to try and results may vary depending on the type of hub used.

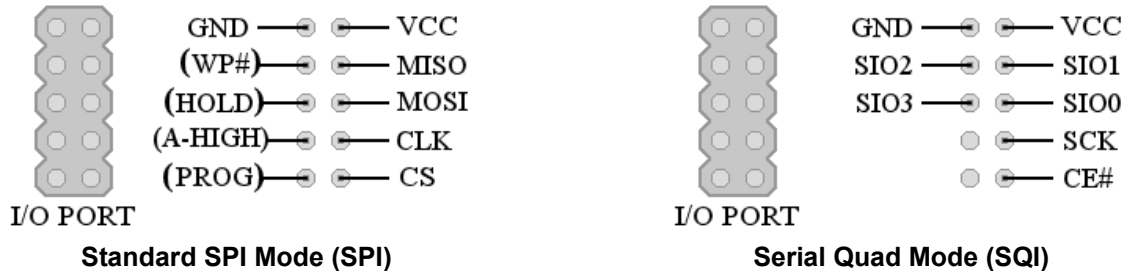
Mountable programming fixtures are available, please contact us to receive a quote.

This feature is supported by all FlashcatUSB products. However, only multiple devices of the same model can be used (you can not mix Classic/Pro for example).

# SPI MODE FOR FLASH PROGRAMMING

If you are using Classic, you will need to make sure the SPI mode firmware is installed.

With the software mode set to SPI NOR or SPI NAND you can use the device as a high-speed programmer for virtually every SPI and SQI compatible Flash memory device.



GND – Ground connections

VCC – Voltage out (can be disabled using SWITCH #1)

MISO – Serial In (receives data from the SPI Flash)

MOSI – Serial Out (sends data to the SPI Flash)

CLK – Serial Clock

CS – Chip-select, also called Chip-Enable.

WP# – Write-protect, some devices require a high-level to enable erasing.

HOLD# – Used to put a device into sleep-mode.

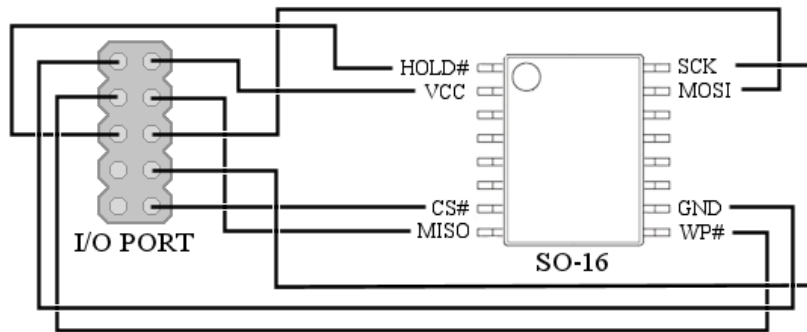
A-HIGH – Active high-level. Use this as an additional pin to set a logic input to high.

PROG – Programmable user pin. You can use the console command: SPI.PROG(1) to set the pin to high-level or SPI.PROG(0) to set it to low-level.

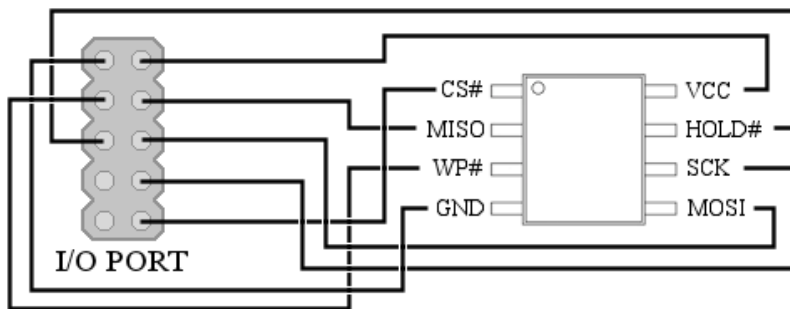
Note: For Nordic nRF24 products, connect the RST pin to (PROG), and the PROG pin on the Nordic device to the A-HIGH pin. When used in this configuration, the PROG pin will automatically pulse (resetting the device) and allowing it to enable SPI Slave mode.

Serial Quad Mode (SQI) supports both dual and quad operation mode.

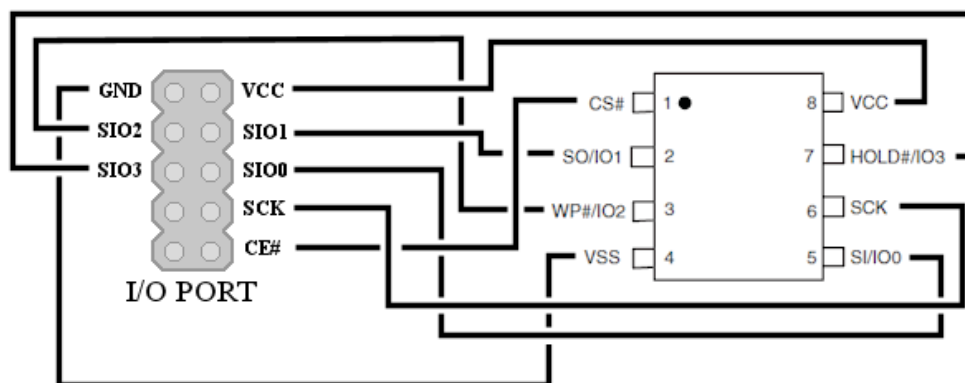
# SPI FLASH WIRING DIAGRAMS



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).





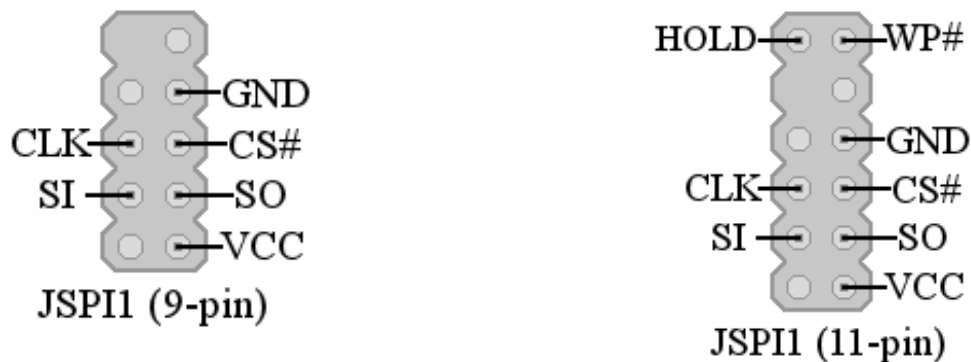
Optionally, you can also purchase drop-in sockets and in-circuit clips for most common SPI chip packages. EmbeddedComputers.net carries many of these in stock.

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.

# PROGRAMING MOTHERBOARDS USING SPI MODE

FlashcatUSB can be used to program the BIOS memory on many motherboards. Newer boards will often use a SO-8 or DIP-8 SPI Flash device, while older boards will often use a PLCC-32 device (which is also compatible with the Extension Port and socket).

Manufacturers often will put a pin header (usually 2mm pin spacing) on the board, so that you can access the memory directly and whilst in-circuit. A PLCC-32 or DIP-8 package memory will usually be in a socket that you will have to manually remove. If the motherboard uses a SO-8 or SO-16 SPI Flash, but does not have a corresponding pin header, then you might be able to use a SOIC test clip to program the memory in-circuit; but be advised, this may not always work, in which case you will have to manually desolder the Flash and use an external socket.

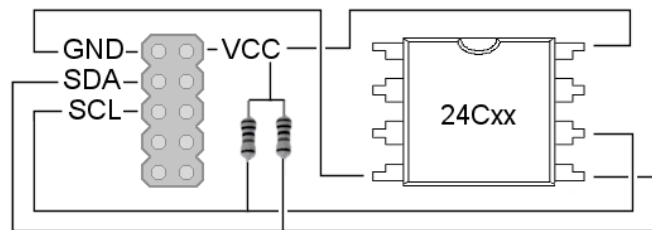


MSI motherboards have a 2mm header labeled "JSPI1". Even though this port is often omitted from the manual, you can connect FlashcatUSB to this header to program the memory. Remember to connect the SO pin from FlashcatUSB to the JSPI1 SI pin, and the SI pin from FlashcatUSB to the JSPI1 SO pin. There are two variants of this header, a 9-pin and a 11-pin. The only difference is with the 11-pin version you will need to connect an additional VCC connection to the WP# pin.

Things to consider: the VCC pin on the FlashcatUSB's pin header is only used to supply power to the target memory device (3.3V @ 120ma). Depending on the motherboard, it may not be possible to power the circuit in order to program the in-circuit memory. In these circumstances, you would need to power the motherboard using it's power supply, while making sure to disconnect and not use the VCC pin from FlashcatUSB.

## HOW 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 newer 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.

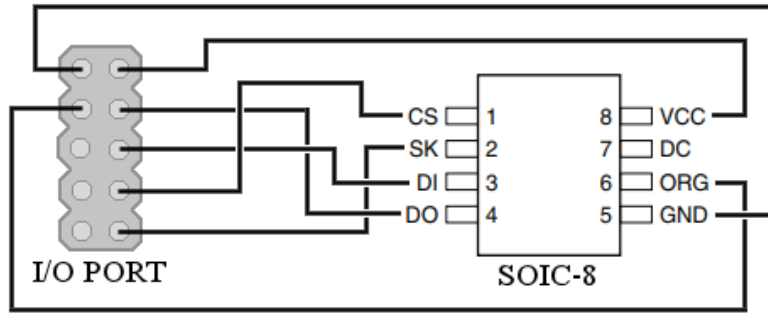
With the device successfully connected, run the software and from Setting menu, select 'I2C EEPROM Mode'. Then click 'Detect' from the 'Main' menu. A new tab will appear that will allow you to read and write to the EEPROM.

Note: For PCB 3.x boards, only PORT B can be used to read/write I2C devices.

# HOW TO PROGRAM MICROWIRE EEPROM

FlashcatUSB supports Microwire 3-wire EEPROM devices. This includes devices that have part numbers that match: 93xx46, 93xx56, 93xx66, 93xx76, 93xx86

Due to the limitations of the 93 series EEPROM devices, automatic detection is not available. So you need to make sure the connection matches the diagram below. If you are using PCB 3.x, you will need to use PORT B.



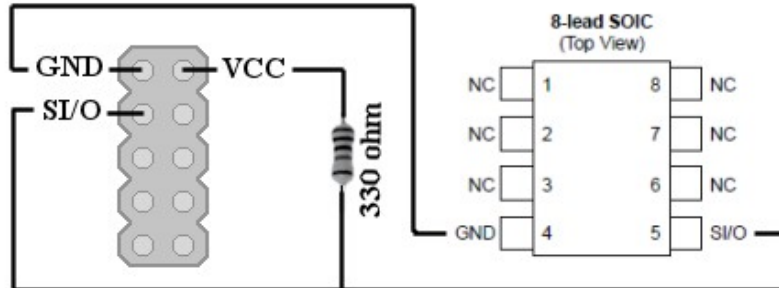
After connecting the device as shown above, open the software and goto the Mode-->Protocol Settings. Then select the MISC tab.

In this section, you need to select your device from the drop down menu and select either X8 or X16 organization.

Close this menu and from the main Settings menu, select the "Microwire EEPROM" mode from the drop down. Finally, click Main-->Detect to have the software apply these settings and attempt to properly connect to this device. Note, if you see all 0s in the Flash tab, this maybe an indication that you are not properly connected to the EEPROM device.

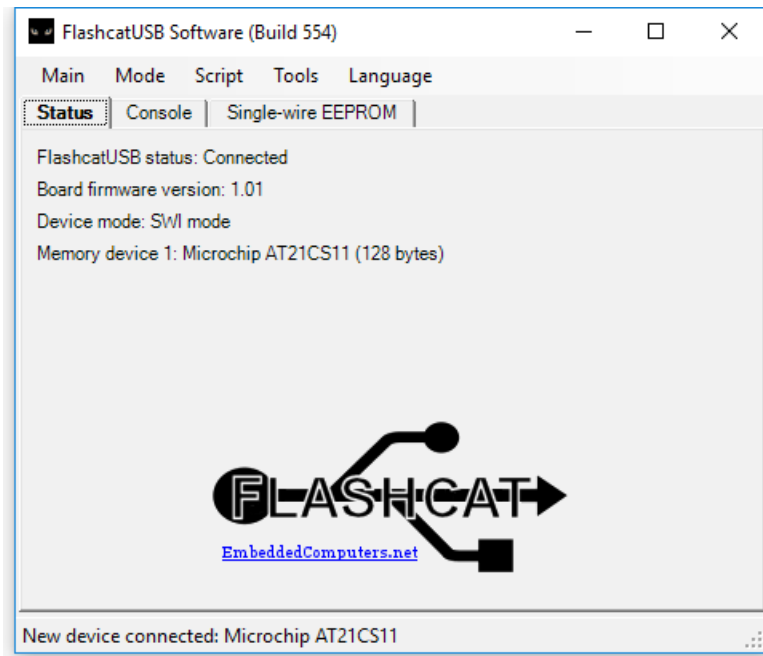
# HOW TO PROGRAM ONE-WIRE EEPROM

FlashcatUSB Classic can be used to program single-wire EEPROMs. For example, the AT21CS01 and AT21CS11. First connect the device to the programmer like this:



You will need to connect a 330 ohm resistor to the SI/O line and the VCC. For VCC make sure the 3.3V switch is set.

Next, make sure your FlashcatUSB has the “FCUSB.CLASSIC.x.xx.SWI” firmware installed. See the bootloader section for information on how to perform a firmware update over USB operation.



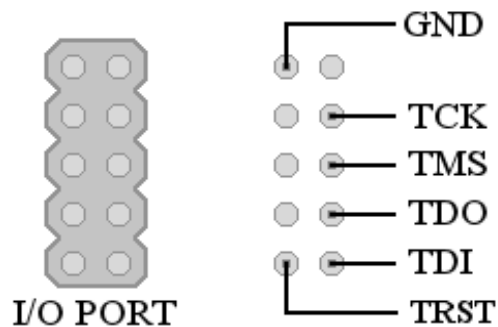
Finally, set the Mode to “1-Wire EEPROM” and the device will automatically be detected.

## JTAG MODE FOR CFI/SPI FLASH PROGRAMMING

FlashcatUSB (Classic and Professional) can be used as a basic JTAG interface for programming memories via a connected Test Access Port (TAP). This is a common method for programming devices that are part of a larger SoC embedded device.

FlashcatUSB Classic requires that the EJTAG firmware be installed (see Bootloader Mode for instructions on how to change AVR firmware). FlashcatUSB Professional uses Port B for JTAG mode.

When the device is in JTAG mode, the software API has full access to a JTAG state-machine, SVF player (for programming CPLD devices), and EJTAG mode (devices that support DMA reading/writing). In addition, SPI over JTAG is supported for Broadcom and Atheros chipsets.



**Classic and Professional 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.

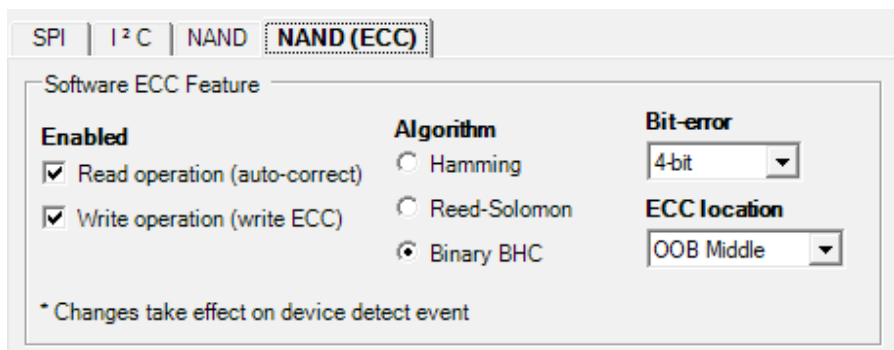
Support for various devices over JTAG is complex. If you need hardware support, please contact us, you maybe required to provide sample equipment as well as a BSDL file.

# SOFTWARE ECC FOR NAND MEMORY

The FlashcatUSB software has support for ECC (Error Correction Code) for SLC NAND devices. This feature is typically needed to successfully read or program NAND memory where bit errors are expected to occur.

The nature of NAND memory and the manufacturing process produces devices with varying levels of unreliability. NAND memory is very unreliable compared to NOR memory. In addition to factory bad blocks (areas of the device that are unusable), data stored on NAND memory will randomly change over time. To overcome this limitation, the spare area of the device is often used to store ECC data that can be used to automatically correct it.

However, different devices require different types of ECC implementations and there is no industry standard as to the types of ECC used or the location to store the ECC data. So this means you will need to configure the software to be able to properly use this feature.



To enable this feature, open up the Settings tab (Mode-->Protocol Settings) and go to the NAND (ECC) tab. Here you can select to use the feature on Read operations and/or Write operations.

When *Read operation* is enabled, the software will automatically try to read the ECC data from the spare page and using the selected algorithm and bit-error rate, and then attempt to parse and correct the data as it being read from the device.

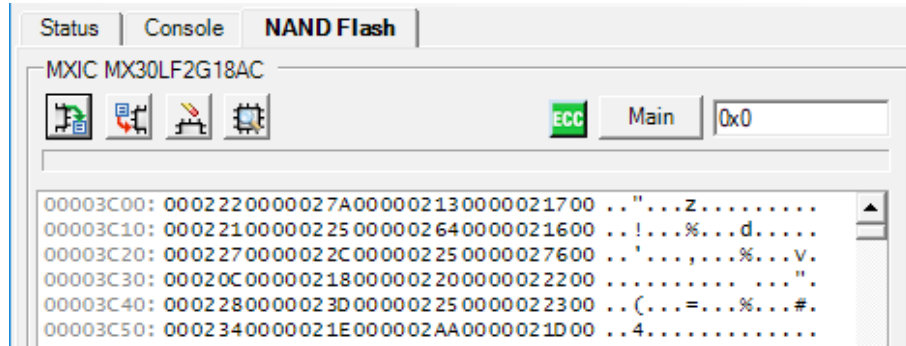
When *Write operation* is enabled, the software will automatically calculate the ECC data and write it to the specified location in the Spare area.

The *ECC location* is used to specify where the ECC data is stored.

**OOB Middle** means to put the data in the middle of the Spare area data designated for each 512-byte block. For example, a typical NAND device that has 2048 byte pages with 64 byte spare area will generate four ECC data segments that will be stored at Spare area offset 0x8, 0x18, 0x28 and 0x38.

**OOB End** means to put the data at the end (aligned to the right) of the Spare area data designated for each 512-byte block. The starting offset will depend on the number of bytes the ECC data creates.

Any change to this tab will require that you reload the Flash device using the "Main-->Detect".

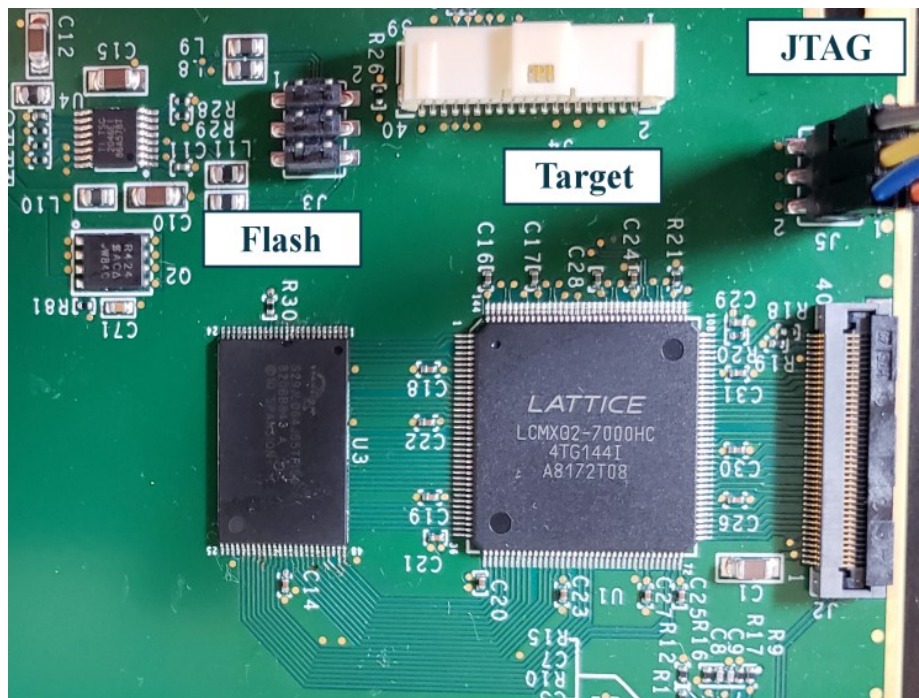


When this feature is enabled for Read operation, the NAND Flash tab can be used to see this feature in action. The icon next to the area selector button will appear. When this icon is GREEN, it means that the software is able to correctly load the ECC data from the Spare area and that the data being show has been corrected (if needed). If this icon is BLUE, it means ECC data was not found and therefore is unable to auto-correct the read operation.



# USING BOUNDARY SCAN PROGRAMMER

FlashcatUSB Professional can be used to program a parallel NOR flash that is connected to a MCU, CPLD, or FPGA with a JTAG interface. This can be accomplished by using a little-known process called 'boundary scan programming', where the JTAG host controller puts the target device into an interconnect test (EXTEST) mode and continually updates all of the pins on the device to "simulate" the NOR programming interface.



The benefits of using this feature is that the external hardware does not need any special configuration or specific JTAG registers in order to program the memory and is considered the most versatile and universal method. However there are drawbacks. First, data transfer speeds are going to be slow. For example, with the JTAG clock set to 20MHz you should expect a read speed of 5KB/s and a write speed of 2.5KB/s.

To use this feature, first you need to obtain the 'Target' BSDL file. This is a plain-text file that describes all of the physical pins for a particular IC and all of the internal pins and registers.

In the BSDL file, look for the line "attribute BOUNDARY\_LENGTH" and at the end of this it will say "entity is <NUMBER>"; This number is the number of bits that the boundary scan register is.

Next, the file will then list all of the bits in the boundary scan register, starting with the bit index and then several parameter. The second parameter will be the pin description. Using this, you need to make a map of all the pins that connect to the Flash.

A typical NOR flash will have address pins, data pins, and control pins. You need to correctly create a list of all the pin indexes that are associated with the NOR flash pins.

For example, a 2mbit bootrom will have 20 address pins (labeled AD0 to AD19), 16 data pins (labeled DQ0 to DQ16) and 6 control pins labeled CE# (chip-enable), OE# (output-enable), WE# (write-enable), BYTE# (Byte/Word select), WP# (write-protect), and RESET#. RB# is also listed but you can ignore that as its not used. Some PCB designs may hardwire CE# to GND and RESET# and WP# to VCC, in which case you can ignore those too. But if they are wired to the Target IC, you need to include them. Note: datasheets will often vary on the pin terminology, some might use G instead of OE, A0 instead of AD, D0 instead of DQ0, etc.

With a map of all the TAGET IC pins to the NOR FLASH pins, you need to create a text file (and name it with the .FCS extension indicating it is a FlashcatUSB script file) and put the following script commands:

```
BoundaryScan.Setup(664)
BoundaryScan.AddPin(331, "DQ0")
BoundaryScan.AddPin(259, "DQ1")
....
BoundaryScan.Detect()
```

The **BoundaryScan.Setup** command inputs the total number of bits of the boundary scan register (BSR). This is the number you got early from the BSDL file.

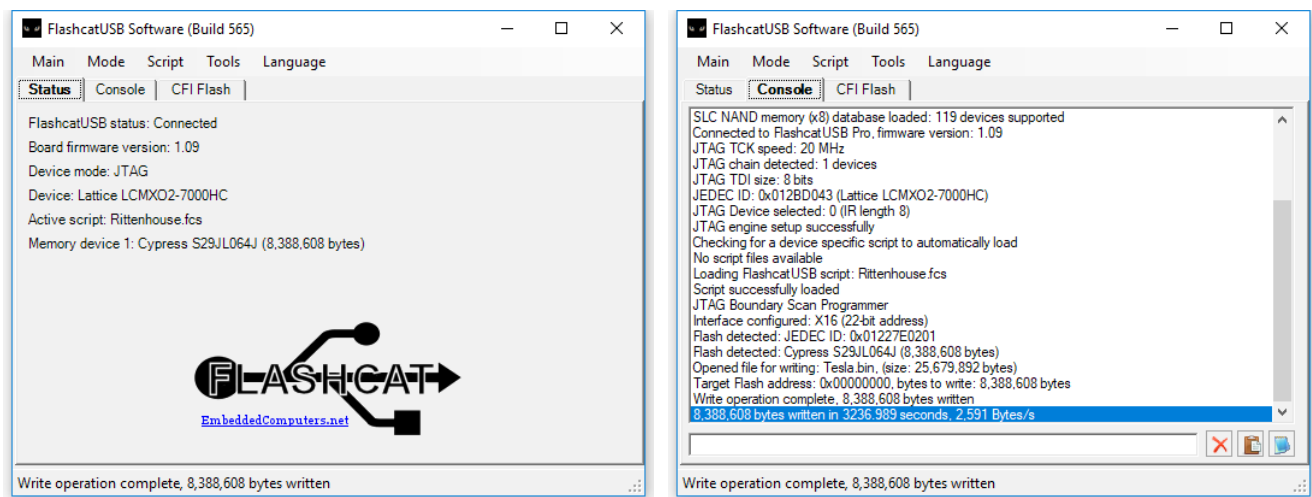
The **BoundaryScan.AppPin** command specifies which BSR bit is associated with each NOR FLASH pin. Make sure you use these pin terminology:

Data pins: "DQ0" – "DQ15" (note: for X8 devices, only specify up to DQ7)

Address pins: "AD0" – "AD25" (note: you only need specify up to the number of address pins you will use)

Control pins: "WE#", "OE#", "BYTE#", "WP#", "RESET#", "CE#"

Finally the **BoundaryScan.Detect** command will initiate the software and attempt to detect the NOR flash, if it is successful, you will see this:



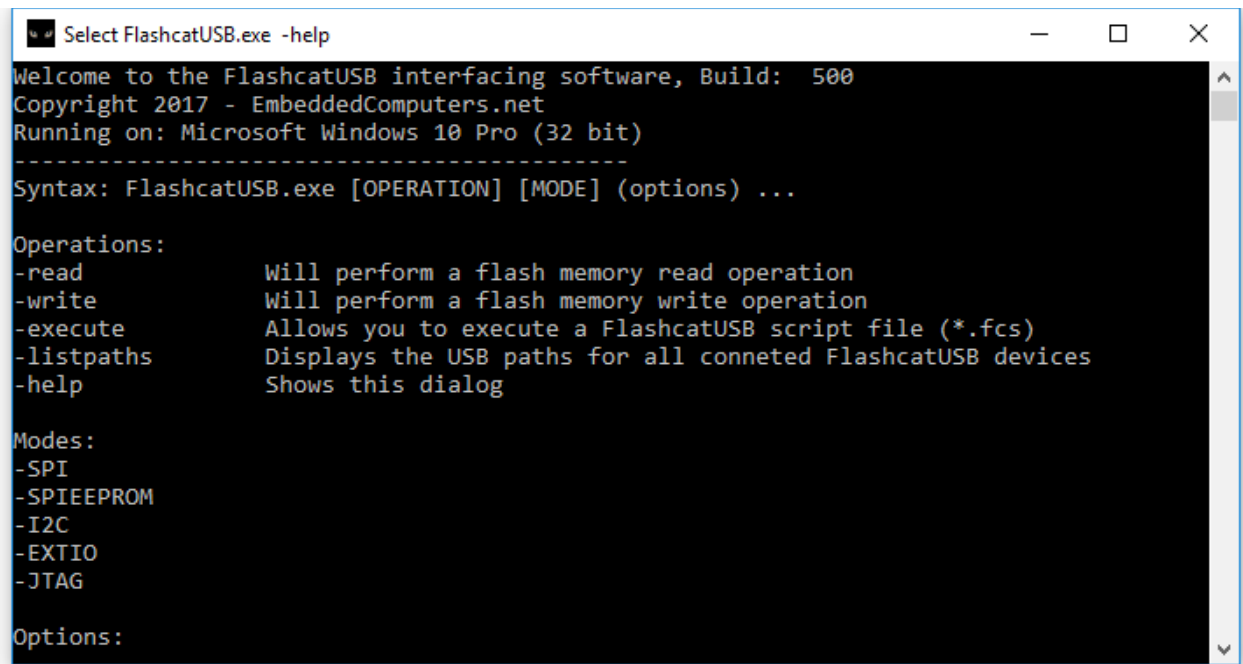
Engineering support is available for this feature, please contact us via email if you need to setup a support account with your company.

## 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.



```
Select FlashcatUSB.exe -help
Welcome to the FlashcatUSB interfacing software, Build: 500
Copyright 2017 - EmbeddedComputers.net
Running on: Microsoft Windows 10 Pro (32 bit)
-----
Syntax: FlashcatUSB.exe [OPERATION] [MODE] (options) ...

Operations:
-read          Will perform a flash memory read operation
-write        Will perform a flash memory write operation
-execute      Allows you to execute a FlashcatUSB script file (*.fcs)
-listpaths    Displays the USB paths for all conneted FlashcatUSB devices
-help         Shows this dialog

Modes:
-SPI
-SPIEEPROM
-I2C
-EXTIO
-JTAG

Options:
```

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. The second parameter is used to specify the protocol (SPI/I2C/EXTIO). Additional parameters are then added to specify certain settings.

For example, to use the console to read an entire SPI Flash memory:

```
FlashcatUSB.exe -read -spi -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).

# 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. Note: this list is updated quarterly, the software may already support newer devices.

## Serial Peripheral Interface Flash Devices (SPI):

AT45DB641E (64Mbit)	S70FL01GS (1Gbit)	MT25QL02GC (2Gbit)
AT45DB642D (64Mbit)	S25FL512S (512Mbit)	N25Q00A (1Gbit)
AT45DB321E (32Mbit)	S70FL256P (256Mbit)	N25Q512A (512Mbit)
AT45DB321D (32Mbit)	S25FL256S (256Mbit)	N25Q256A (256Mbit)
AT45DB161E (16Mbit)	S25FL256S (256Mbit)	N25Q256A (256Mbit)
AT45DB161D (16Mbit)	S25FL128P (128Mbit)	NP5Q128A (128Mbit)
AT45DB081E (8Mbit)	S25FL128P (128Mbit)	N25Q128 (128Mbit)
AT45DB081D (8Mbit)	S25FL129P (128Mbit)	N25Q064A (64Mbit)
AT45DB041E (4Mbit)	S25FL129P (128Mbit)	N25Q064 (64Mbit)
AT45DB041D (4Mbit)	FL127S/FL128S (128Mbit)	N25Q032 (32Mbit)
AT45DB021E (2Mbit)	S25FL128S (128Mbit)	N25Q016 (16Mbit)
AT45DB021D (2Mbit)	S25FL127S (128Mbit)	N25Q008 (8Mbit)
AT45DB011D (1Mbit)	S25FL064L (64Mbit)	M25P128 (128Mbit)
AT25DF641 (64Mbit)	S25FL064 (64Mbit)	M25P64 (64Mbit)
AT25DF321S (32Mbit)	S25FL032 (32Mbit)	M25PX32 (32Mbit)
AT25DF321 (32Mbit)	S25FL016A (16Mbit)	M25P32 (32Mbit)
AT25DF161 (16Mbit)	S25FL008A (8Mbit)	M25PX16 (16Mbit)
AT25DF081 (8Mbit)	S25FL040A (4Mbit)	M25P16 (16Mbit)
AT25DF021 (2Mbit)	S25FL164K (64Mbit)	M25P80 (8Mbit)
AT26DF321 (32Mbit)	S25FL132K (32Mbit)	M25PX80 (8Mbit)
AT26DF161 (16Mbit)	S25FL216K (16Mbit)	M25P40 (4Mbit)
AT26DF161A (16Mbit)	S25FL116K (16Mbit)	M25P20 (2Mbit)
AT26DF081A (8Mbit)	S25FL208K (8Mbit)	M25P10 (1Mbit)
AT25SL321 (32Mbit)	S25FL204K (4Mbit)	M25P05 (64Kbit)
AT25SF321 (32Mbit)	S25FL004A (4Mbit)	M25PX64 (64Mbit)
AT25SF161 (16Mbit)	MX25L51245G (512Mbit)	M25PX32 (32Mbit)
AT25SF081 (8Mbit)	MX25L25655E (256Mbit)	M25PX16 (16Mbit)
AT25SF041 (4Mbit)	MX25L256 (256Mbit)	M25PE16 (16Mbit)
AT25XV041 (4Mbit)	MX25L12855E (128Mbit)	M25PE80 (8Mbit)
AT25XV021 (2Mbit)	MX25L128 (128Mbit)	M25PE40 (4Mbit)
W25M512JV (512Mbit)	MX25U12873F (128Mbit)	M25PE20 (2Mbit)
W25Q512 (512Mbit)	MX25R6435 (64Mbit)	M25PE10 (1Mbit)
W25Q256JV (256Mbit)	MX25L6455E (64Mbit)	M45PE16 (16Mbit)
W25Q256 (256Mbit)	MX25L640 (64Mbit)	M45PE80 (8Mbit)
W25Q128JV (128Mbit)	MX25L320 (32Mbit)	M45PE40 (4Mbit)

W25Q128 (128Mbit)	MX25L3205D (32Mbit)	M45PE20 (2Mbit)
W25Q64 (64Mbit)	MX25L323 (32Mbit)	M45PE10 (1Mbit)
W25Q32 (32Mbit)	MX25L3255E (32Mbit)	EN25Q128 (128Mbit)
W25Q16 (16Mbit)	MX25U3235F (32Mbit)	EN25Q32 (32Mbit)
W25Q80 (8Mbit)	MX25R3235F (32Mbit)	EN25Q16 (16Mbit)
W25Q80BW (8Mbit)	MX25L1633E (16Mbit)	EN25Q80 (8Mbit)
W25Q40 (4Mbit)	MX25L160 (16Mbit)	EN25Q40 (4Mbit)
W25Q128FW (128Mbit)	MX25L80 (8Mbit)	EN25QH128 (128Mbit)
W25Q64FW (64Mbit)	MX25L40 (4Mbit)	EN25QH64 (64Mbit)
W25Q32FW (32Mbit)	MX25L20 (2Mbit)	EN25QH32 (32Mbit)
W25Q16FW (16Mbit)	MX25L10 (1Mbit)	EN25QH16 (16Mbit)
W25Q08EW (8Mbit)	MX25U643 (64Mbit)	EN25QH80 (8Mbit)
W25X64 (64Mbit)	MX25U323 (32Mbit)	EN25P64 (64Mbit)
W25X64 (64Mbit)	MX25U163 (16Mbit)	EN25P32 (32Mbit)
W25X32 (32Mbit)	MX25U803 (8Mbit)	EN25P16 (16Mbit)
W25X16 (16Mbit)	MX25L512 (64Kbit)	EN25F32 (32Mbit)
W25X80 (8Mbit)	MX25L1021E (1Mbit)	EN25F16 (16Mbit)
W25X40 (4Mbit)	MX25L5121E (64Kbit)	EN25F80 (8Mbit)
W25X20 (2Mbit)	MX66L51235F (512Mbit)	EN25F40 (4Mbit)
W25X10 (2Mbit)	MX25V8035 (8Mbit)	EN25F20 (2Mbit)
W25X05 (1Mbit)	MX25V4035 (4Mbit)	EN25T32 (32Mbit)
SST26VF064B (64Mbit)	MX25V8035F (8Mbit)	EN25T16 (16Mbit)
SST26VF064 (64Mbit)	PM25LV016B (16Mbit)	EN25T80 (8Mbit)
SST26VF032 (32Mbit)	PM25LV080B (8Mbit)	EN25T40 (4Mbit)
SST26VF032B (32Mbit)	PM25LV040 (4Mbit)	EN25T20 (2Mbit)
SST26WF032 (32Mbit)	PM25LV020 (2Mbit)	GD25Q128 (128Mbit)
SST26VF016 (16Mbit)	PM25LV010 (1Mbit)	GD25Q64 (64Mbit)
SST26VF032 (32Mbit)	PM25LV512 (64Kbit)	GD25Q32 (32Mbit)
SST26VF016B (16Mbit)	PM25LD020 (2Mbit)	GD25Q16 (16Mbit)
SST26VF016 (16Mbit)	Pm25LD010 (1Mbit)	GD25Q80 (8Mbit)
SST26WF016B (16Mbit)	Pm25LD512 (64Kbit)	GD25Q40 (4Mbit)
SST26WF080B (8Mbit)	A25LQ64 (64Mbit)	GD25Q20 (2Mbit)
SST26WF040B (4Mbit)	A25LQ32A (32Mbit)	GD25Q10 (1Mbit)
SST25VF128B (128Mbit)	A25L032 (32Mbit)	GD25Q512 (64Kbit)
SST25VF064C (64Mbit)	A25L016 (16Mbit)	GD25VQ16C (16Mbit)
SST25VF032B (32Mbit)	A25LQ16 (16Mbit)	GD25VQ80C (8Mbit)
SST25VF032 (32Mbit)	A25L080 (8Mbit)	GD25VQ41B (4Mbit)
SST25VF016B (16Mbit)	A25L040 (4Mbit)	GD25VQ21B (2Mbit)
SST25VF080B (8Mbit)	A25L020 (2Mbit)	GD25LQ128 (128Mbit)
SST25VF080 (8Mbit)	A25L010 (1Mbit)	GD25LQ64 (64Mbit)
SST25WF080B (8Mbit)	A25L512 (64Kbit)	GD25LQ32 (32Mbit)

SST25PF040C (4Mbit)	A25LS512A (64Kbit)	GD25LQ16 (16Mbit)
SST25WF040B (4Mbit)	FM25Q16A (16Mbit)	GD25LQ80 (8Mbit)
SST25VF040B (4Mbit)	FM25Q32A (32Mbit)	GD25LQ40 (4Mbit)
SST25WF040 (4Mbit)	FM25M04A (4Mbit)	GD25LQ20 (2Mbit)
SST25WF020A (2Mbit)	FM25M08A (8Mbit)	GD25LQ10 (1Mbit)
SST25LF020A (2Mbit)	FM25M16A (16Mbit)	IS25CD020 (2Mbit)
SST25WF020 (2Mbit)	FM25M32A (32Mbit)	IS25CD010 (1Mbit)
SST25VF020 (2Mbit)	FM25M64A (64Mbit)	IS25CD512 (64Kbit)
SST25WF010 (1Mbit)	FM25M4AA (4Mbit)	IS25CD025 (32Kbit)
SST25VF010 (1Mbit)	AT25128B (16Kbit)	IS25CQ032 (32Mbit)
SST25WF512 (64Kbit)	AT25256B (32Kbit)	IS25LP256 (256Mbit)
SST25VF512 (64Kbit)	AT25512 (64Kbit)	IS25LP128 (128Mbit)
SST25VF020A (2Mbit)	AT25010A (0.125Kbit)	IS25LP064 (64Mbit)
SST25VF010A (1Mbit)	AT25020A (0.25Kbit)	IS25LP032 (32Mbit)
25AA160A (2Kbit)	AT25040A (0.5Kbit)	IS25LP016 (16Mbit)
25AA160B (2Kbit)	AT25080 (1Kbit)	IS25LP080 (8Mbit)
F25L04 (4Mbit)	AT25160 (2Kbit)	IS25LQ032 (32Mbit)
F25L04 (4Mbit)	AT25320 (4Kbit)	IS25LQ016 (16Mbit)
F25L08 (8Mbit)	AT25640 (8Kbit)	IS25LQ080 (8Mbit)
F25L08 (8Mbit)	M95010 (0.125Kbit)	IS25LQ040 (4Mbit)
F25L32QA (32Mbit)	M95020 (0.25Kbit)	IS25LQ020 (2Mbit)
LE25FU406B (4Mbit)	M95040 (0.5Kbit)	IS25LQ010 (1Mbit)
BG25Q32A (32Mbit)	M95080 (1Kbit)	IS25LQ512 (64Kbit)
M95M01 (1Mbit)	M95160 (2Kbit)	IS25LQ025 (32Kbit)
M95M02 (2Mbit)	M95320 (4Kbit)	IS25LD040 (4Mbit)
IS25WQ040 (4Mbit)	M95640 (8Kbit)	IS25WD040 (4Mbit)
IS25WQ020 (2Mbit)	M95128 (16Kbit)	IS25WD020 (2Mbit)
IS25WP040 (4Mbit)	M95256 (32Kbit)	IS25WP256 (256Mbit)
IS25WP020 (2Mbit)	M95512 (64Kbit)	IS25WP128 (128Mbit)
IS25WP080 (8Mbit)	IS25WP016 (16Mbit)	IS25WP064 (64Mbit)
IS25WP032 (32Mbit)		

## Serial NAND Devices (SPI-NAND):

MT29F1G01ABA (1Gbit)	GD5F1GQ4UB (1Gbit)	W25M02GV (2Gbit)
MT29F1G01ABB (1Gbit)	GD5F1GQ4RB (1Gbit)	W25M02GW (2Gbit)
MT29F2G01AAA (2Gbit)	GD5F1GQ4UC (1Gbit)	W25N01GV (1Gbit)
MT29F2G01ABA (2Gbit)	GD5F1GQ4RC (1Gbit)	W25N01GW (1Gbit)
MT29F2G01ABB (2Gbit)	GD5F2GQ4UB (2Gbit)	W25N512GV (512Mbit)
MT29F4G01ADA (4Gbit)	GD5F2GQ4RB (2Gbit)	W25N512GW (512Mbit)
MT29F4G01AAA (4Gbit)	GD5F2GQ4UC (2Gbit)	GD5F2GQ4RC (2Gbit)
GD5F4GQ4UA (4Gbit)	GD5F4GQ4UB (4Gbit)	GD5F4GQ4RB (4Gbit)
GD5F4GQ4UC (4Gbit)	GD5F4GQ4RC (4Gbit)	

### Parallel NOR Devices (Parallel Flash NOR/NAND):

28F320J3 (32Mbit)	AM29F010B (1Mbit)	39SF512 (64Kbit)
28F640J3 (64Mbit)	AM29F040B (4Mbit)	39SF010 (1Mbit)
28F128J3 (128Mbit)	AM29LV200(T) (2Mbit)	39SF020 (2Mbit)
28F256J3 (256Mbit)	AM29LV200(B) (2Mbit)	39LF010 (1Mbit)
28F320J5 (32Mbit)	AM29F200(T) (2Mbit)	39LF020 (2Mbit)
28F640J5 (64Mbit)	AM29F200(B) (2Mbit)	39LF040 (4Mbit)
28F800C3(T) (8Mbit)	AM29LV400(T) (4Mbit)	39VF800 (8Mbit)
28F800C3(B) (8Mbit)	AM29LV400(B) (4Mbit)	39VF160 (16Mbit)
28F160C3(T) (16Mbit)	AM29F400(T) (4Mbit)	39VF1681 (16Mbit)
28F160C3(B) (16Mbit)	AM29F400(B) (4Mbit)	39VF1682 (16Mbit)
28F320C3(T) (32Mbit)	AM29LV800(T) (8Mbit)	39VF1601 (16Mbit)
28F320C3(B) (32Mbit)	AM29LV800(B) (8Mbit)	39VF1602 (16Mbit)
28F640C3(T) (64Mbit)	AM29F800(T) (8Mbit)	39VF1602C (16Mbit)
28F640C3(B) (64Mbit)	AM29F800(B) (8Mbit)	39VF3201 (32Mbit)
28F400B3(T) (4Mbit)	AM29LV160B(T) (16Mbit)	39VF3202 (32Mbit)
28F400B3(B) (4Mbit)	AM29LV160B(B) (16Mbit)	39VF6401 (64Mbit)
28F800B3(T) (8Mbit)	AM29DL322G(T) (32Mbit)	39VF6402 (64Mbit)
28F800B3(B) (8Mbit)	AM29DL322G(B) (32Mbit)	MX29L3211 (32Mbit)
28F160B3(T) (16Mbit)	AM29DL323G(T) (32Mbit)	MX29LV040 (4Mbit)
28F160B3(B) (16Mbit)	AM29DL323G(B) (32Mbit)	MX29LV400T (4Mbit)
28F320B3(T) (32Mbit)	AM29DL324G(T) (32Mbit)	MX29LV400B (4Mbit)
28F320B3(B) (32Mbit)	AM29DL324G(B) (32Mbit)	MX29LV800T (8Mbit)
28F640B3(T) (64Mbit)	AM29LV320D(T) (32Mbit)	MX29LV800B (8Mbit)
28F640B3(B) (64Mbit)	AM29LV320D(B) (32Mbit)	MX29LV160DT (16Mbit)
AT29C010A (1Mbit)	AM29LV320M(T) (32Mbit)	MX29LV160DB (16Mbit)
AT49F512 (64Kbit)	AM29LV320M(B) (32Mbit)	MX29LV320T (32Mbit)
AT49F010 (1Mbit)	M29W800AT (8Mbit)	MX29LV320B (32Mbit)
AT49F020 (2Mbit)	M29W800AB (8Mbit)	MX29LV640ET (64Mbit)
AT49F040 (4Mbit)	M28W160CT (16Mbit)	MX29LV640EB (64Mbit)
AT49F040T (4Mbit)	M28W160CB (16Mbit)	MX29GL128F (128Mbit)

AT49BV/LV16X (16Mbit)	M29W160ET (16Mbit)	M29F200FT (2Mbit)
AT49BV/LV16XT (16Mbit)	M29W160EB (16Mbit)	M29F200FB (2Mbit)
S29GL128P (128Mbit)	M29D323DT (32Mbit)	M29F400FT (4Mbit)
S29GL256P (256Mbit)	M29D323DB (32Mbit)	M29F400FB (4Mbit)
S29GL512P (512Mbit)	M28W320FCT (32Mbit)	M29F800FT (8Mbit)
S29GL01GP (1Gbit)	M28W320FCB (32Mbit)	M29F800FB (8Mbit)
S29JL064J (64Mbit)	M28W320BT (32Mbit)	M29F160FT (16Mbit)
S29GL032M (32Mbit)	M28W320BB (32Mbit)	M29F160FB (16Mbit)
S29GL032M (32Mbit)	M29W320DT (32Mbit)	M29W160ET (16Mbit)
S29GL032M(B) (32Mbit)	M29W320DB (32Mbit)	M29W160EB (16Mbit)
S29GL032M(T) (32Mbit)	M28W640ECT (64Mbit)	M29W320DT (32Mbit)
S29GL064M (64Mbit)	M28W640ECB (64Mbit)	M29W320DB (32Mbit)
S29GL064M (64Mbit)	M58LW064D (64Mbit)	M29W640GH (64Mbit)
S29GL064M(T) (64Mbit)	K8P1615UQB (16Mbit)	M29W640GL (64Mbit)
S29GL064M(B) (64Mbit)	K8D1716UT (16Mbit)	M29W640GT (64Mbit)
S29GL064M (64Mbit)	K8D1716UB (16Mbit)	M29W640GB (64Mbit)
S29GL128M (128Mbit)	K8D3216UT (32Mbit)	M29W128GH (128Mbit)
S29GL256M (256Mbit)	K8D3216UB (32Mbit)	M29W128GL (128Mbit)
S29GL128 (128Mbit)	K8P3215UQB (32Mbit)	M29W256GH (256Mbit)
S29GL256 (256Mbit)	K8D6316UT (64Mbit)	M29W256GL (256Mbit)
S29GL512 (512Mbit)	K8D6316UB (64Mbit)	M29W512G (512Mbit)
S29GL01G (1Gbit)	K8P6415UQB (64Mbit)	MBM29LV200TC (2Mbit)
S70GL02G (2Gbit)	K8P2716UZC (128Mbit)	MBM29LV200BC (2Mbit)
TC58FVT800 (8Mbit)	K8Q2815UQB (128Mbit)	MBM29LV400TC (4Mbit)
TC58FVB800 (8Mbit)	K8P5516UZB (256Mbit)	MBM29LV400BC (4Mbit)
TC58FVT160 (16Mbit)	K8P5615UQA (256Mbit)	MBM29LV800TA (8Mbit)
TC58FVB160 (16Mbit)	HY29F400T (4Mbit)	MBM29LV800BA (8Mbit)
TC58FVT321 (32Mbit)	HY29F400B (4Mbit)	MBM29LV160T (16Mbit)
TC58FVB321 (32Mbit)	HY29F800T (8Mbit)	MBM29LV160B (16Mbit)
W49F002U (2Mbit)	HY29F800B (8Mbit)	MBM29LV320TE (32Mbit)
W29EE512 (64Kbit)	HY29LV400T (4Mbit)	MBM29LV320BE (32Mbit)
W29C010 (1Mbit)	HY29LV400B (4Mbit)	MBM29DL32XTD (32Mbit)
W29C020 (2Mbit)	HY29LV800T (8Mbit)	MBM29DL32XBD (32Mbit)
W29C040 (4Mbit)	HY29LV800B (8Mbit)	LH28F160S3 (16Mbit)
W29GL032CT (32Mbit)	HY29LV160T (16Mbit)	LH28F320S3 (32Mbit)
W29GL032CB (32Mbit)	HY29LV160B (16Mbit)	LH28F160BJE (16Mbit)
LHF00L15 (32Mbit)	HY29LV320T (32Mbit)	LH28F320BJE (32Mbit)
	HY29LV320B (32Mbit)	

## SLC NAND Devices (Parallel Flash NOR/NAND):



NAND128W3A (128Mbit)	TC58DVM92A5TA10 (512Mbit)	MX30LF1208AA (512Mbit)
NAND256R3A (256Mbit)	TC58NVG0S3HTA00 (1Gbit)	MX30LF1GE8AB (1Gbit)
NAND256W3A (256Mbit)	TC58NVG0S3HTA10 (1Gbit)	MX30UF1G18AC (1Gbit)
NAND512R3A (512Mbit)	TC58NVG1S3HTA00 (2Gbit)	MX30LF1G18AC (1Gbit)
NAND512W3A (512Mbit)	TC58NVG1S3HTA10 (2Gbit)	MX30LF1G08AA (1Gbit)
NAND01GR3A (1Gbit)	TC58NVG2S0HTA00 (4Gbit)	MX30LF2G18AC (2Gbit)
NAND01GW3A (1Gbit)	TC58NVG2S0HTA10 (4Gbit)	MX30UF2G18AC (2Gbit)
NAND04GW3B (4Gbit)	TH58NVG3S0HTA00 (8Gbit)	MX30LF2G28AB (2Gbit)
MT29F2G08AAB (2Gbit)	TH58NVG3S0HTA10 (8Gbit)	MX30LF2GE8AB (2Gbit)
MT29F4G08BAB (4Gbit)	W29N01GV (1Gbit)	MX30UF2G18AB (2Gbit)
MT29F1G08ABAEA (1Gbit)	W29N02GV (2Gbit)	MX30UF2G28AB (2Gbit)
MT29F1G08ABBEA (1Gbit)	HY27SS08561A (256Mbit)	MX30LF4G18AC (4Gbit)
MT29F1G08ABADAWP (1Gbit)	HY27US08561A (256Mbit)	MX30UF4G18AB (4Gbit)
MT29F2G08ABBFA (2Gbit)	HY27US0812(1/2)B (512Mbit)	MX30LF4G28AB (4Gbit)
MT29F2G08ABAFB (2Gbit)	H27U1G8F2B (1Gbit)	MX30LF4GE8AB (4Gbit)
MT29F4G08AAA (4Gbit)	HY27UF081G2M (1Gbit)	MX30UF4G28AB (4Gbit)
MT29F8G08BAA (8Gbit)	HY27US081G1M (1Gbit)	MX60LF8G18AC (8Gbit)
K9F1G08U0D (1Gbit)	HY27SF081G2M (1Gbit)	MX60LF8G28AB (8Gbit)
K9F1G08U0B (1Gbit)	HY27UF082G2B (2Gbit)	S34ML01G1 (1Gbit)
K9F1G08X0 (1Gbit)	HY27UF082G2A (2Gbit)	S34ML02G1 (2Gbit)
K9F1G08U0E (1Gbit)	A5U1GA31ATS (1Gbit)	S34ML04G1 (4Gbit)
K9F2G08X0 (2Gbit)	K9F2G08U0M (4Gbit)	S34ML01G2 (1Gbit)
K9F2G08U0C (2Gbit)	S34ML04G2 (4Gbit)	S34ML02G2 (2Gbit)

### EEPROM devices (SPI EEPROM):

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)		

### HyperFlash devices (HyperFlash):

Cypress S26KS128S  
Cypress S26KL128S

Cypress S26KS256S  
Cypress S26KL256S

Cypress S26KS512S  
Cypress S26KL512S

**MCU Devices with internal programmable memory (SPI):**

Nordic nRF24LE1  
Nordic nRF24LU1+  
Altera EPCS128

Altera EPCS1  
Altera EPCS16

Altera EPCS4  
Altera EPCS64

**MCU Devices with internal programmable memory (JTAG):**

Xilinx XC2C32A  
Xilinx XC2C256  
Xilinx XC9500XL  
Xilinx XC9572XL  
Lattice LC4064V  
Lattice LCMXO1200

Xilinx XC2C64A  
Xilinx XC2C384  
Xilinx XC95288XL  
Xilinx XC9536XL  
Lattice LCMXO256  
Lattice LCMXO2280

Xilinx XC2C128  
Xilinx XC2C512  
Xilinx XC95144XL  
Lattice LC4032V  
Lattice LCMXO640