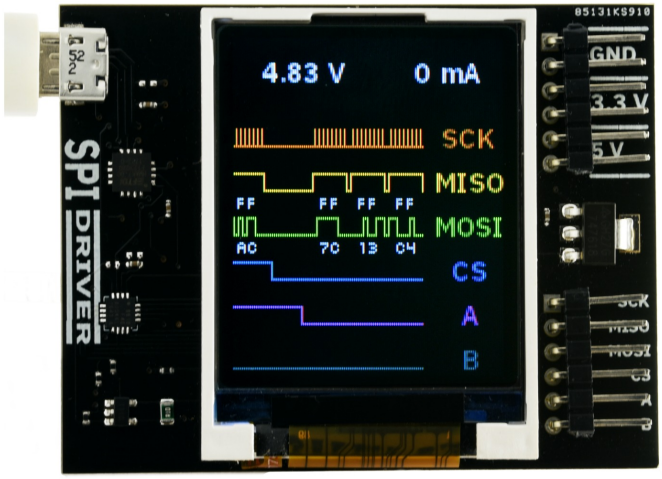


Contents

1 Overview

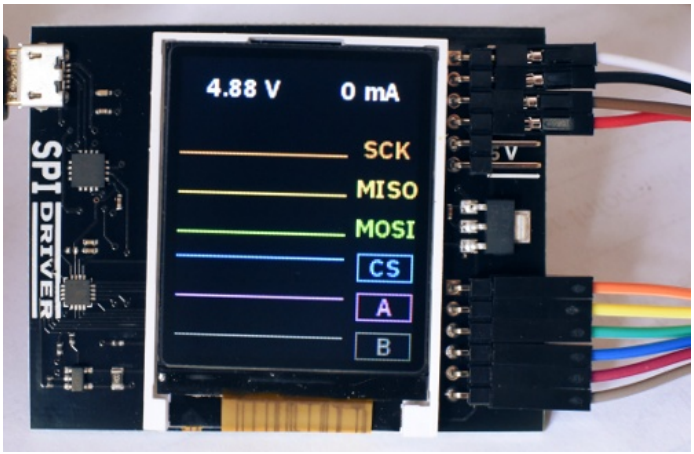
SPIDriver is an easy-to-use tool for controlling SPI devices over USB. It works with Windows, Mac, and Linux, and has a built-in color screen that shows a live logic-analyzer display of all SPI traffic.

1.1 Features

- live display shows you exactly what its doing all the time
- sustained SPI transfers at 500 Kbps
- USB line voltage monitor to detect supply problems, to 0.01 V
- target device high-side current measurement, to 5 mA
- two auxiliary output signals, A and B
- dedicated power out lines. two each of GND, 3.3 V and 5 V
- all signals color coded to match jumper colors
- all signals are 3.3 V, and are 5 V tolerant
- uses an FTDI USB serial adapter, and Silicon Labs automotive-grade EFM8 controller
- also reports uptime, temperature, and running CRC of all traffic
- all sensors and signals controlled using a simple serial protocol
- GUI, command-line, C/C++, and Python 2/3 host software provided for Windows, Mac, and Linux

2 Getting Started

When you first connect it to the USB port, the display blinks white for a moment then shows something like this:



Connect the six colored hookup jumpers as shown, following the same sequence as on the screen

SCK	orange
MISO	yellow
MOSI	green
CS	blue
A	purple
B	gray

The top six signals carry power. There are two each of GND, 3.3 V and 5 V.

Across the top of the display SPIDriver continuously measures the USB input voltage, and the current output.

3 Software installation

The source for all the SPIDriver software is the repository. Available are:

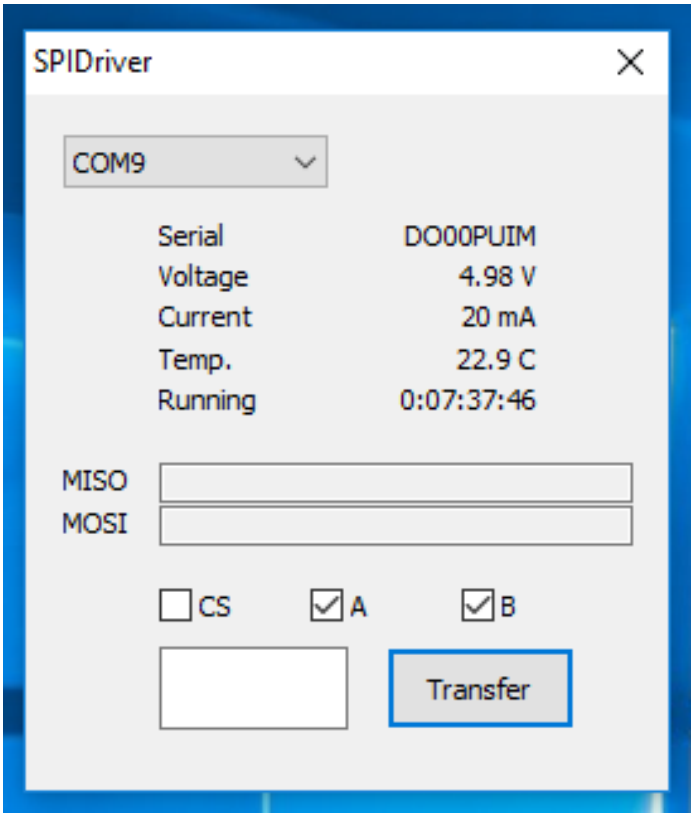
- a Windows/Mac/Linux GUI
- a Windows/Mac/Linux command-line
- Python 2 and 3 bindings
- Windows/Mac/Linux C/C++ bindings

Installation of the GUI and command-line utilities varies by platform.

3.1 Windows

This installer contains the GUI and command-line utilities. The GUI shortcut is installed on the desktop:

launching it brings up the control window:



If there is only one serial device, the SPIDriver device should be automatically selected. If there is more than one device, select its COM port from the pull-down menu at the top. Once connected, you can control the signal lines and transmit hex values.

The command line utility `spicl` is also installed. For example to display status information:

```
C:\>"c:\Program Files\Excamera Labs\SPIDriver\spicl.exe" COM3 i
uptime 1625 4.810 V 45 mA 23.3 C
```

See below for more information on the command-line syntax.

3.2 Linux

The Linux GUI is available for download as spigui-linux64. Alternatively you can run the native Python GUI directly as shown below.

To build the command-line tool, clone the repository, then do:

```
$ cd spidriver/c
$ make -f linux/Makefile
$ ./build/spicl /dev/ttyUSB0 i
```

and you should see something like:

```
uptime 2285 4.812 V 45 mA 23.6 C
```

3.3 MacOS

The MacOS GUI is available for download as spigui-macos. This is a Mac executable, so after downloading it do:

```
$ cd Downloads
$ chmod a+x spigui-macos
$ ./spigui-macos
```

Alternatively you can run the native Python GUI directly as shown below.

To build the command-line tool, clone the repository, then do:

```
cd spidriver/c
make -f linux/Makefile
./build/spicl /dev/cu.usbserial-DO00QS8D i
```

(substituting your actual SPIDriver's ID for `DO00QS8D`) and you should see something like:

```
uptime 2285 4.812 V 45 mA 23.6 C
```

Note that the port to use is `/dev/cu.usbserial-XXXXXXXX`, as explained here.

3.4 Python 2 and 3

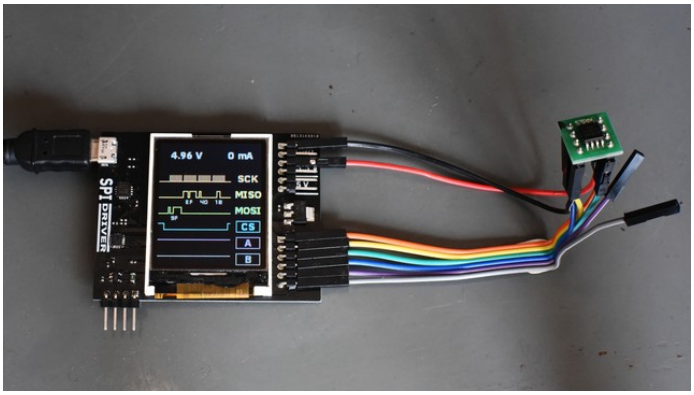
The SPIDriver bindings can be installed with `pip` like this:

```
pip install spidriver
```

then from Python you can do:

```
>>> from spidriver import SPIDriver
>>> s = SPIDriver("/dev/ttyUSB0") # change for your port
>>> s.sel()                    # start command
>>> s.write([0x9f])            # command 9F is READ JEDEC ID
>>> list(s.read(3))            # read next 3 bytes
[239, 64, 24]
>>> s.unsel()                  # end command
>>>
```

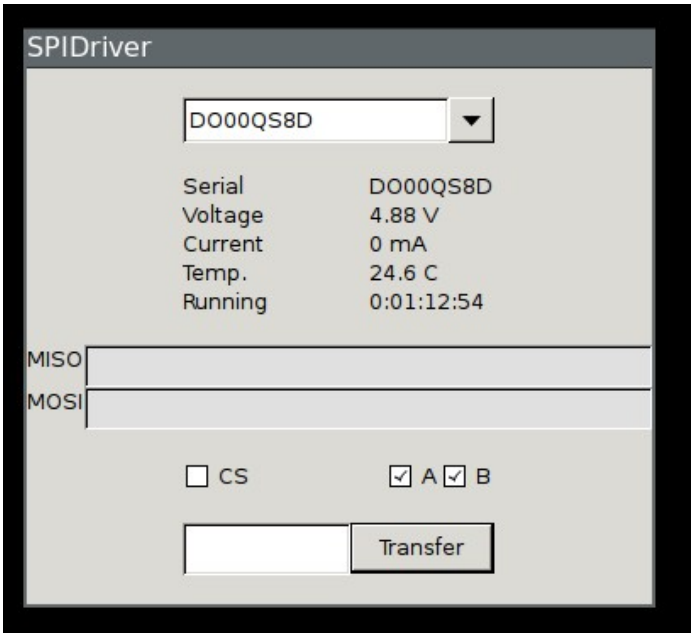
and you should see something like:



The GUI (which uses wxPython) can be run with:

```
python spigui.py
```

which depending on your distribution looks something like this:



There are more examples in the `python/samples` folder in the repository.

3.5 C/C++

SPIDriver is contained in a single source file with a single header. Both are in the `c/common` folder in the repository. Usage follows the Python API and is fairly self-explanatory.

3.6 The command-line tool `spicl`

`spicl` is the same on all platforms.

The first parameter to the command is the serial port, which depends on your operating system. All following parameters are control commands. These are:

`i`	print status information (uptime, voltage, current, temperature)
`s`	SPI select
`u`	SPI unselect
`w` *byte*,...	write bytes to SPI
`r N`	read N bytes from SPI
`a 0/1`	Set A line
`b 0/1`	Set B line

For example the command:

```
spicl /dev/ttyUSB0 s w 0x9f r 3 u
```

does the following:

`s`	SPI select
`w 0x9f`	Write byte `0x9f`
`r 3`	Read 3 bytes from slave
`u`	SPI unselect

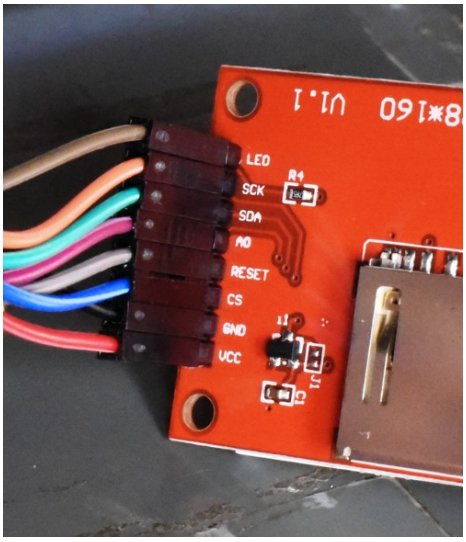
Byte values are decimal by default, or hex if prefixed with `0x`. To send multiple bytes, separate them with a comma.

4 Using SPIDriver

4.1 ST7735R 1.8" LCD

The ST7735R is a small color LCD controller, with a panel resolution of 160x128. It has an SPI-like interface. It does not use MISO, and needs an extra control line to distinguish between command bytes and data bytes. It requires an external reset for reliable start-up.

Connect the LCD panel like this:



SPIDriver		LCD
3.3V	brown	LED
SCK	orange	SCK
MOSI	green	SDA
A	purple	A0
B	gray	RESET
CS	blue	CS
GND	black	GND
3.3V	red	VCC

There is a Python demonstration program for loading images onto the display. It requires pillow for image loading, so you may need to install its prerequisites:

```
pip install pillow
```

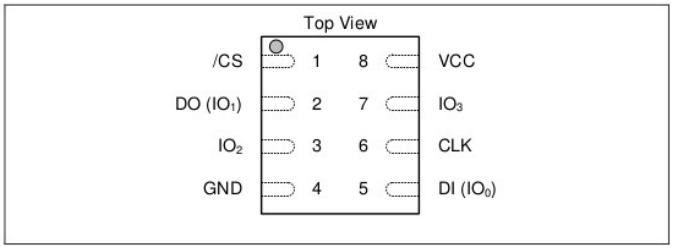
Then in the SPIDriver repository do:

```
cd python/samples
python st7735s.py -h /dev/ttyUSB0 grace.png
```

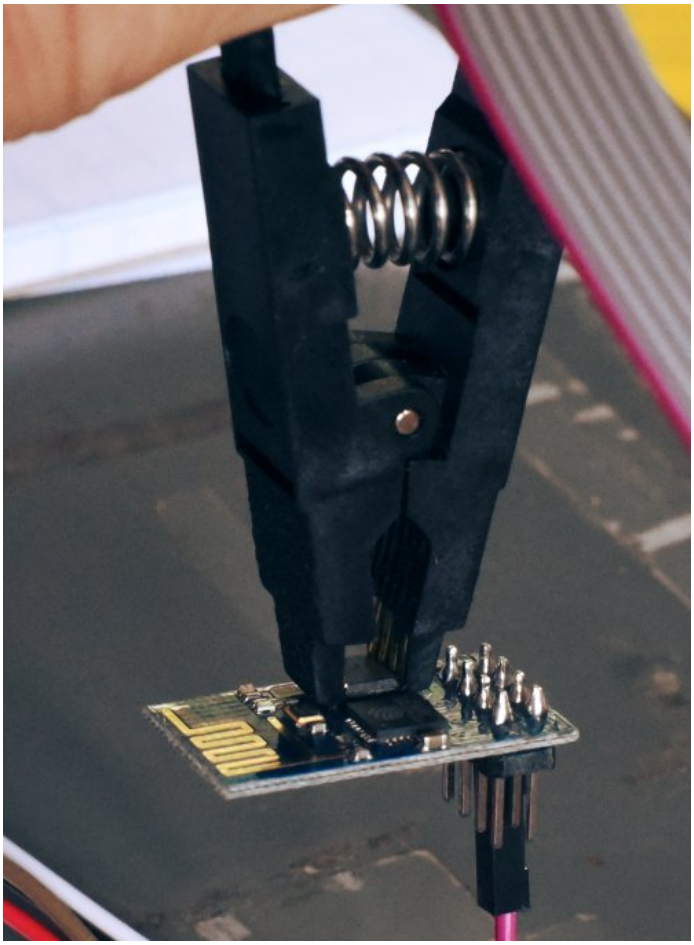
You can supply any image files as the last argument, and the script will resize and display them in order.

4.2 SPI Flash

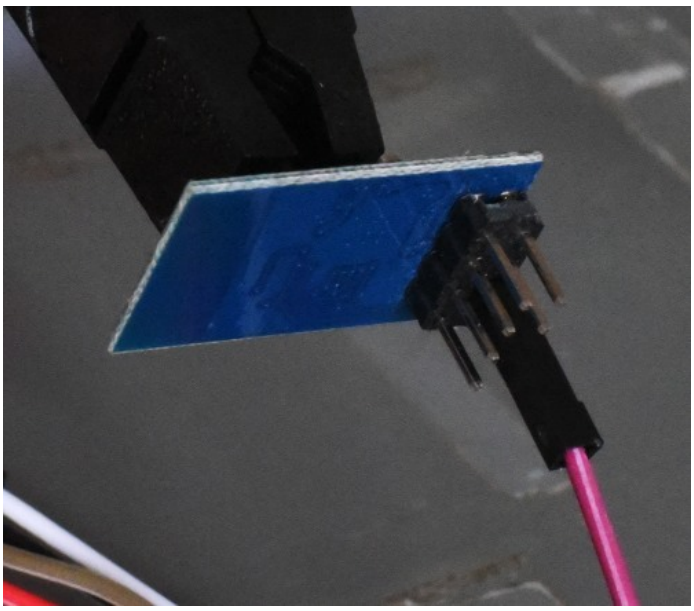
First confirm the flash pinout on its datasheet. Modern SPI flash chips have a pinout like this one:

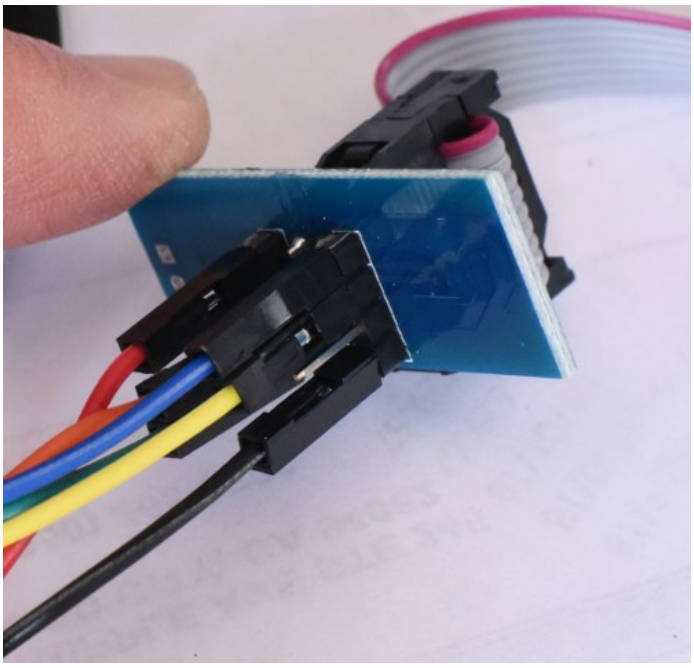


Connect the clip to the flash, oriented with the red cable on pin 1. For example this is the clip connected to the flash on a ESP8266 board:



For the ESP8266, the CPU needs to be kept in reset so there is no contention while SPIDriver is driving to the flash. Connect SPIDriver's A signal (purple) to the ESP8266 reset:





and connect the clip cable pins to the SPIDriver like this:

SPIDriver		flash
CS	blue	1
MISO	yellow	2
		3
GND	black	4
MOSI	green	5
SCK	orange	6
		7
3.3V	red	8

You should be able to use the command-line utility to confirm the flash 3-byte JEDEC ID. For example:

```
$ spicl /dev/ttyUSB0 a 0 u s w 0x9f r 3 u
0xc8,0x40,0x13
```

The specific ID codes vary by manufacturer, and are listed in the datasheet for each flash part. The third byte is the flash size in bits, so the flash above is $2^{19} = 524288$ bytes, or 512K bytes. If you don't see a valid ID, and you've checked the pinout, make sure that the clip is properly making contact with all of the flash pins.

There is a Python example flash.py that can read and write flash contents using standard SPI flash commands. With no options, it prints out JEDEC ID as confirmation:

```
$ python flash.py -h /dev/ttyUSB0
Got JEDEC ID: c8 40 13
Flash size is 524288 bytes
```

You can read all of flash into a file with the `-r` option:

```
$ python flash.py -h /dev/ttyUSB0 -r flashfile
Got JEDEC ID: c8 40 13
```

```
Flash size is 524288 bytes
0/512 KBytes
8/512 KBytes
...
504/512 KBytes
```

Similarly the `-w` option erases the flash and writes a file to it:

```
$ python flash.py -h /dev/ttyUSB0 -w flashfile
Got JEDEC ID: c8 40 13
Flash size is 524288 bytes
0/512 KBytes
...
504/512 KBytes
```

5 Technical notes

5.1 Port names

The serial port that SPIDriver appears at depends on your operating system.

On **Windows**, it appears as COM1, COM2, COM3 etc. You can use the Device Manager or the `MODE` command to display the available ports. This article describes how to set a device to a fixed port.

On **Linux**, it appears as `/dev/ttyUSB0`, 1, 2 etc. The actual number depends on the order that devices were added. However it also appears as something like:

```
/dev/serial/by-id/usb-FTDI_FT230X_Basic_UART_DO00QS8D-if00-port0
```

Where `DO00QS8D` is the serial code of the SPIDriver (which is printed on the bottom of each SPIDriver). This is longer, of course, but always the same for a given device.

Similarly on **Mac OS**, the SPIDriver appears as `/dev/cu.usbserial-DO00QS8D`.

5.2 Decreasing the USB latency timer

SPIDriver performance can be increased by setting the USB latency timer to its minimum value of 1 ms. This can increase the speed of two-way SPI traffic by up to 10X.

On **Linux** do:

```
setserial /dev/ttyUSB0 low_latency
```

On **Windows** and **Mac OS** follow these instructions.

5.3 Temperature sensor

The temperature sensor is located in the on-board EFM8 microcontroller. It is calibrated at manufacture to within 2 C. A sudden temperature rise may indicate that one of the output pins (MOSI, SCK, CS, A, or B) is shorted to VCC or GND.

5.4 Raw protocol

SPIDriver uses a serial protocol to send and receive SPI commands. Connect to the SPIDriver at 460800 baud, 8 bits, no parity, 1 stop bit (460800 8N1). Many SPIDriver commands are ASCII, you can control it interactively from any terminal application that can connect at 460800 baud. For example typing u and s toggles the CS line and ? displays the status info. Commands are:

`?`	transmit status information (see below)
`e` $byte$	echo $byte$
`s`	select
`u`	unselect
`a` $byte$	set A output to 0/1
`b` $byte$	set B output to 0/1
`x`	disconnect from SPI bus
`0x80-bf`	write and read 1-64 bytes
`0xc0-ff`	write 1-64 bytes

So for example to select, transfer two bytes `0x12,0x34`, unselect, the host sends 5 bytes. The command `0x81` is a two byte send/receive, so two bytes are returned to the PC.

```
s
0x81
0x12
0x34
u
```

The status information is always 80 characters, space padded. For example:

```
[spidriver1 D000QS8D 000007219 4.807 045 25.4 1 1 1 49c1                    ]
```

The fields are space-delimited:

spidriver1	fixed identifier
serial	serial code identifier
uptime	SPIDriver uptime 0-999999999, in seconds
voltage	USB bus voltage, in volts
current	attached device current, in mA
temperature	junction temperature, in C
CS	CS line state
A	A line state
B	B line state
crc	16-bit CRC of all input and output bytes (CRC-16-CCITT)

5.5 Specifications

DC characteristics

	min	typ	max	units
Voltage accuracy		0.01		V
Current accuracy		5		mA
Temperature accuracy		$\pm$ 2		°C
MISO				
low voltage			0.6	V
high voltage	2.7		5.8	V
Output signal current (SCK, MOSI, CS, A, B)			8	mA
Output current			470	mA
Current consumption		25		mA

AC characteristics

	min	typ	max	units
SPI speed	495	500	505	Kbps
Uptime accuracy		150		ppm
Uptime rollover		31.7		years
Startup time			200	ms

6 Support information

Technical and product support is available at support@spidriver.com

SPIDriver is built and maintained by Excamera Labs.