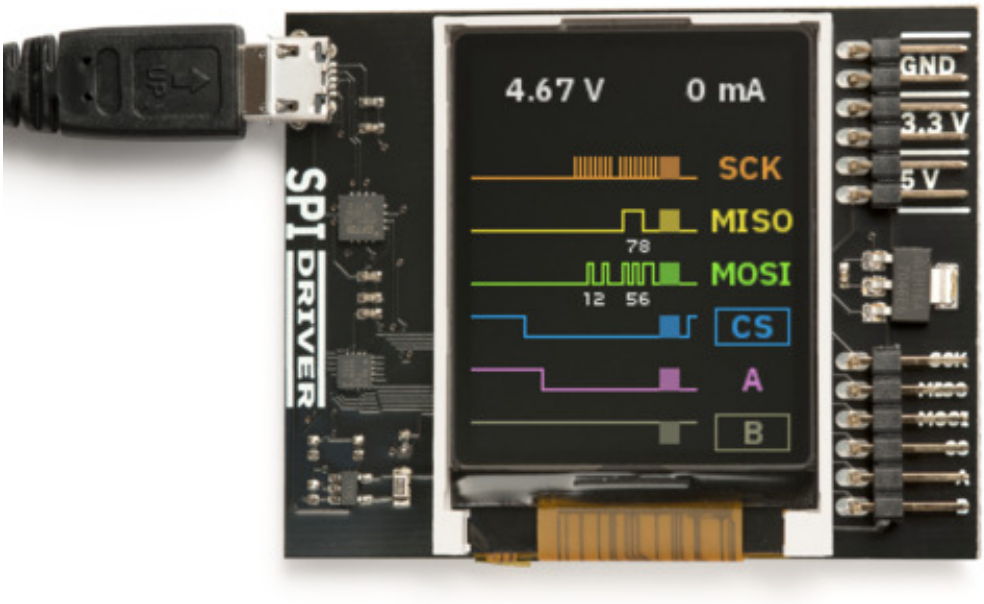
spidriver

James Bowman

Feb 07, 2023

CONTENTS:



SPIDriver is an easy-to-use, open source 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 "dashboard" of all the SPI activity.

The SPIDriver User Guide has complete information on the hardware:

https://spidriver.com/spidriver.pdf

CHAPTER

ONE

SYSTEM REQUIREMENTS

Because it is a pure Python module, `spidriver` can run on any system supported by `pyserial`. This includes:

- Windows 7 or 10
- Mac OS
- Linux, including all Ubuntu distributions

Both Python 2.7 and 3.x are supported.

CHAPTER

TWO

INSTALLATION

The `spidriver` package can be installed from PyPI using `pip`:

```
$ pip install spidriver
```

CHAPTER

THREE

QUICK START

To connect to an SPI flash and read its JEDEC id:

```
>>> 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
```

The User Guide at https://spidriver.com/spidriver.pdf has more examples, as does the SPIDriver repo on github.

CHAPTER

FOUR

MODULE CONTENTS

class spidriver.**SPIDriver**(*port='/dev/ttyUSB0'*)

SPIDriver interface.

Parameters
port (*str*) – The USB port to connect to

After connection, the following object variables reflect the current values of the SPIDriver. They are updated by calling *getstatus()*.

Variables

- **product** – product code e.g. 'spidriver1' or 'spidriver2'
- **serial** – serial string of SPIDriver
- **uptime** – time since SPIDriver boot, in seconds
- **voltage** – USB voltage, in V
- **current** – current used by attached device, in mA
- **temp** – temperature, in degrees C
- **cs** – state of CS pin
- **a** – state of A pin
- **b** – state of B pin
- **ccitt_crc** – CCITT-16 CRC of all transmitted and received bytes

detach()

Detach all signals, leaving them all to float.

sel()

Select the SPI device by asserting CS

unsel()

Unselect the SPI device by deasserting CS

read(*l*)

Read l bytes from the SPI device

Parameters
l (*int*) – number of bytes to read

Return bytes
received bytes, length l

`write`(*bb*)

Write bb to the SPI device

Parameters
bb (*bytes*) – bytes to write to the SPI device

`writeread`(*bb*)

Write bytes to the SPI device, return the read bytes

Parameters
bb (*bytes*) – bytes to write to the SPI device

Return bytes
received bytes, same length as bb

`seta`(*v*)

Set the A signal to 0 or 1

`setb`(*v*)

Set the B signal to 0 or 1

`setmode`(*m*)

Set the SPI mode to 0,1,2 or 3

`getstatus`()

Update all status variables

INDEX