SCaLE 2019 - Introduction to SPI/SPIDEV

Introduction to SPI/SPIDEV

Tom King

Stephanie Lockwood-Childs

https://cm.e-ale.org/2019/SCaLE17x/spidev/SCaLE-2019-SPI_SPIDEV.pdf

SCaLE 2019 - Introduction to SPI/SPIDEV

Brought to you by:

THE LINUX FOUNDATION

VCT

- Linux Foundation Training has provided speaker funding

What is SPI?

- Serial Peripheral Interface
- Motorola
- de facto standard
- master-slave bus
- 4 wire bus
    - except when it's not
- no maximum clock speed
- "A glorified shift register"

http://wikipedia.org/wiki/Serial_Peripheral_Interface

SCaLE 2019 - Introduction to SPI/SPIDEV

Common uses of SPI

- Flash memory
- ADCs
- Chromium Embedded Controller
- LCD Controllers
- Sensors
  - Thermocouples and other high data rate devices

SCaLE 2019 - Introduction to SPI/SPIDEV

Advantages:

- Full Duplex in default mode
- Uses 4 pins (or 3 in some implementations)
- Low Processor overhead (even bit banged)
- No “unique address” needed (often just setting a GPIO pin to address)
- No “Protocol” to decode. (although can be used as transport for Protocols)

SCaLE 2019 - Introduction to SPI/SPIDEV

Disadvantages:

- Higher pin count than i2c
- No in-band addressing (need HW pins to address)
- No slave ack that the data/command got to the intended recipient.
- No error checking
- Relatively short distances (often only onboard)

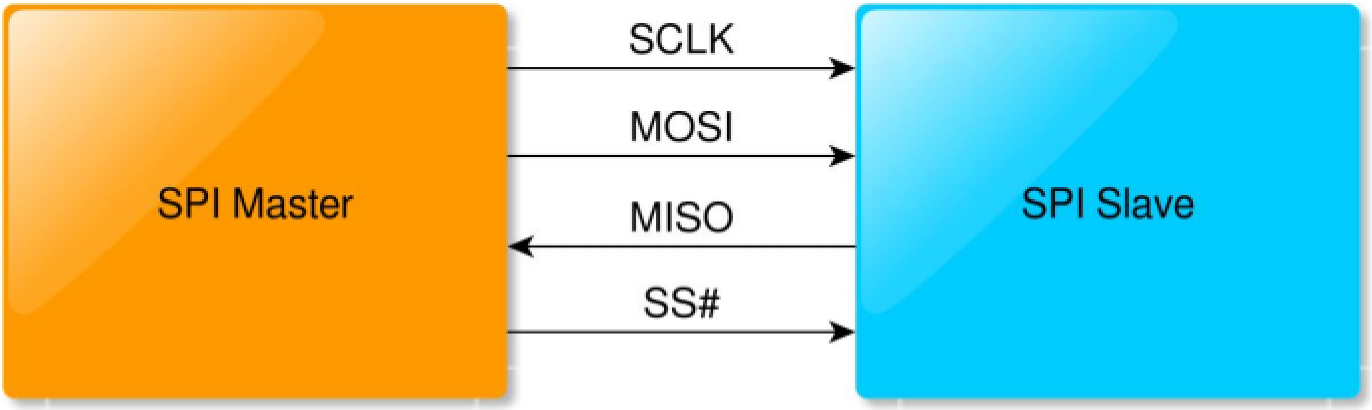
SCaLE 2019 - Introduction to SPI/SPIDEV

SPI Signals

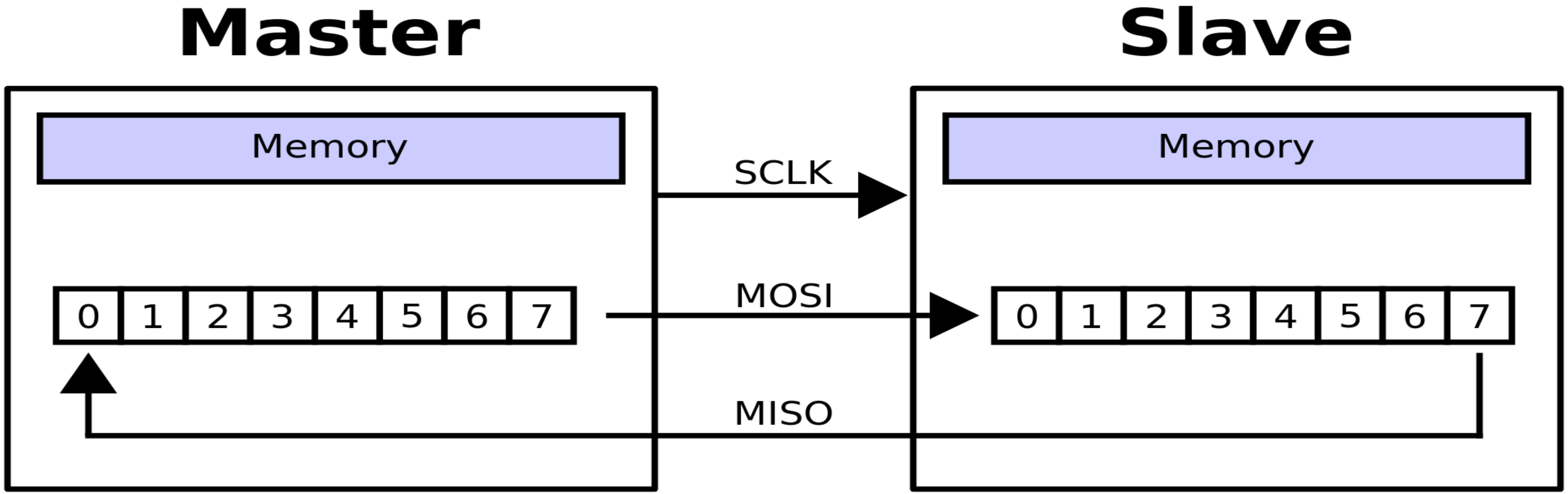
- MOSI - Master Output Slave Input
  - SIMO, SDI, DI, SDA
- MISO - Master Input Slave Output
  - SOMI, SDO, DO, SDA
- SCLK - Serial Clock (Master output)
  - SCK, CLK, SCL
- $\overline{SS}$ - Slave Select (Master output)
- CSn, EN, ENB

SCaLE 2019 - Introduction to SPI/SPIDEV

SPI Master and Slave

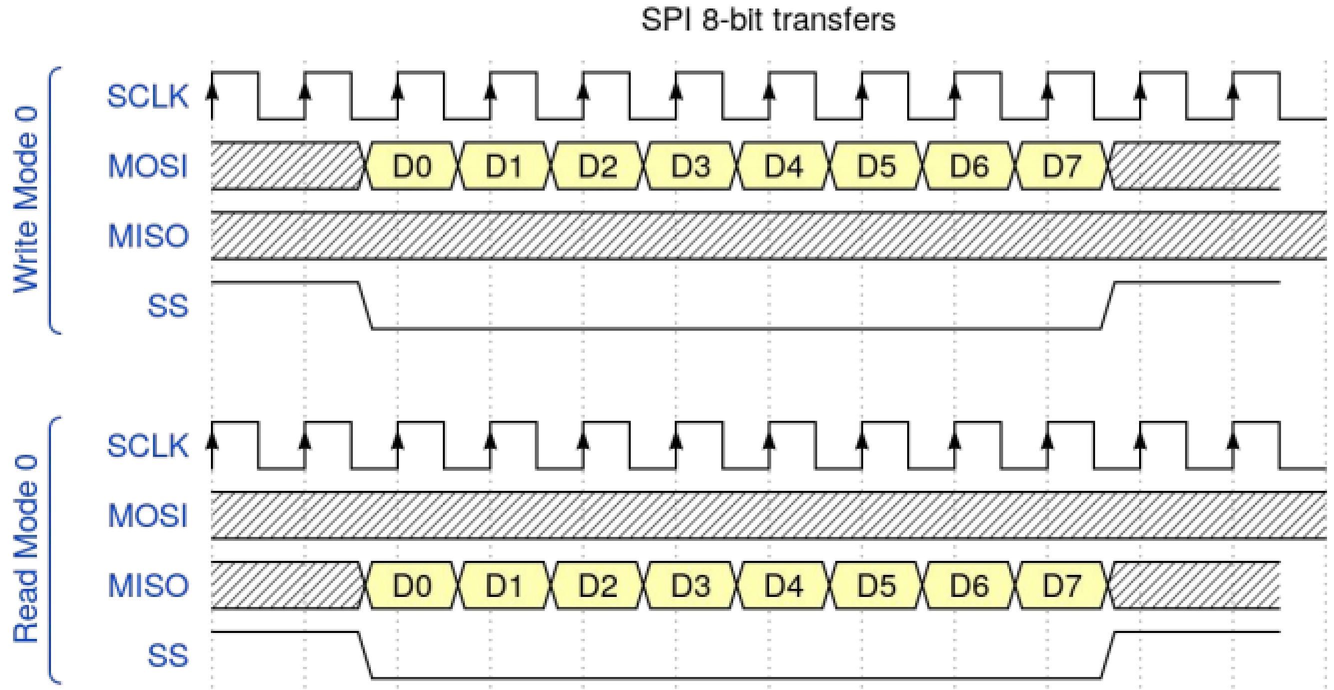


SCaLE 2019 - Introduction to SPI/SPIDEV



SCaLE 2019 - Introduction to SPI/SPIDEV

Basic SPI Timing Diagram



SCaLE 2019 - Introduction to SPI/SPIDEV

SPI Modes

- Modes are composed of two clock characteristics
- CPOL - clock polarity
  - 0 = clock idle state low
  - 1 = clock idle state high
- CPHA - clock phase
  - 0 = data latched falling, output rising
  - 1 = data latched rising, output falling

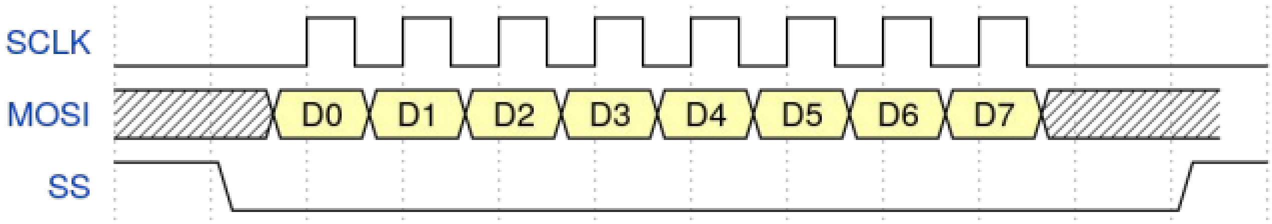
SCaLE 2019 - Introduction to SPI/SPIDEV

SPI Modes Cont'd

Mode	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1

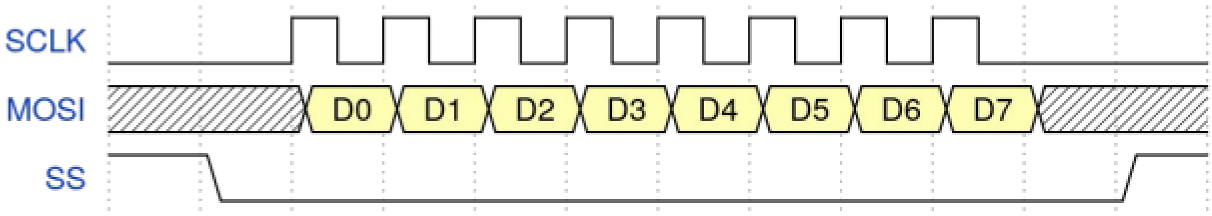
SPI Mode Timing - CPOL 0

SPI Write Mode 0



Clock idle low, data latched on rising edge

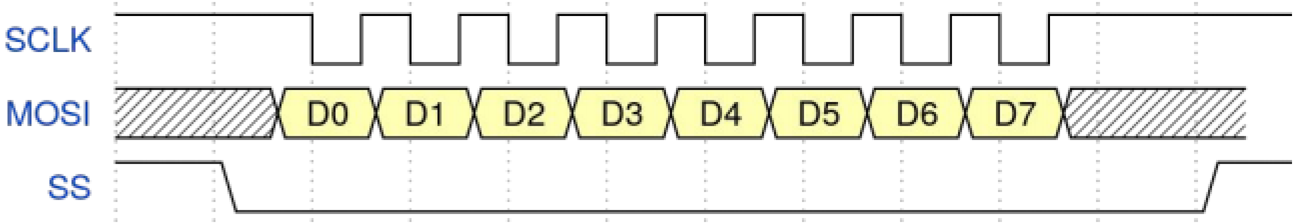
SPI Write Mode 1



Clock idle low, data latched on falling edge

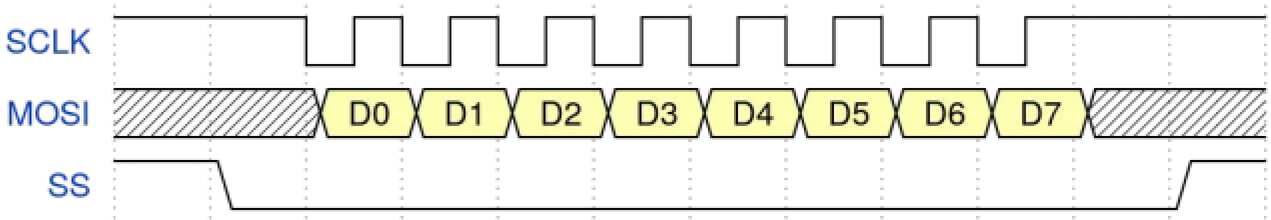
SPI Mode Timing - CPOL 1

SPI Write Mode 2



Clock idle high, data latched on falling edge

SPI Write Mode 3



Clock idle high, data latched on rising edge

SCaLE 2019 - Introduction to SPI/SPIDEV

Let’s look at an example together:

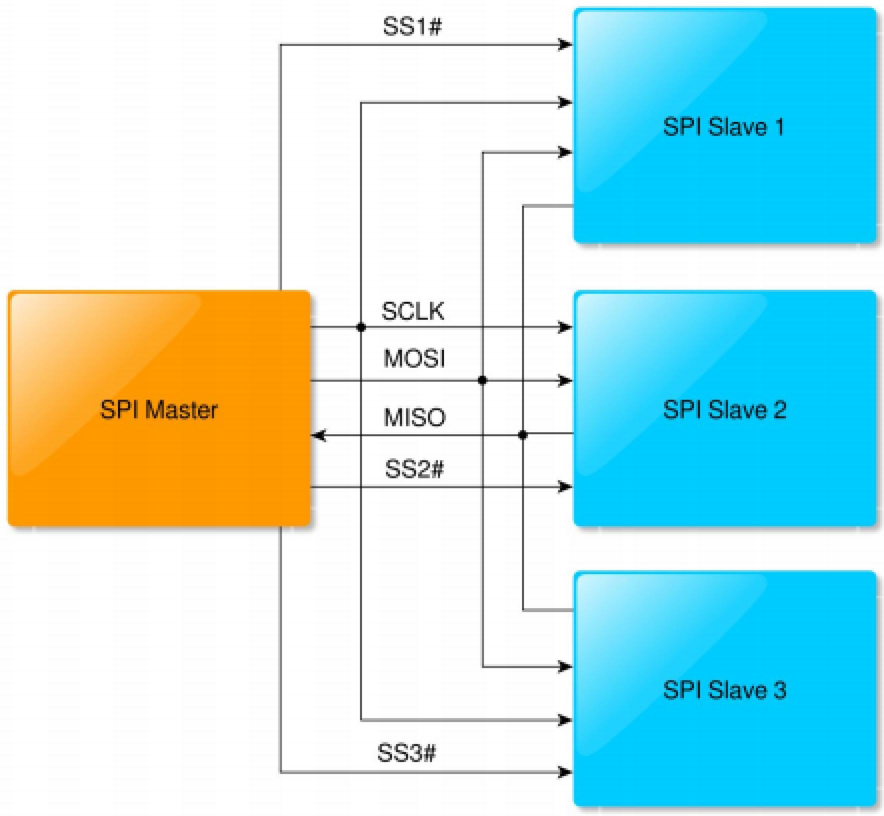
https://en.wikipedia.org/wiki/Serial_Peripheral_Interface#Example_of_bit-banging_the_master_protocol

SPI <u>can</u> be more complicated

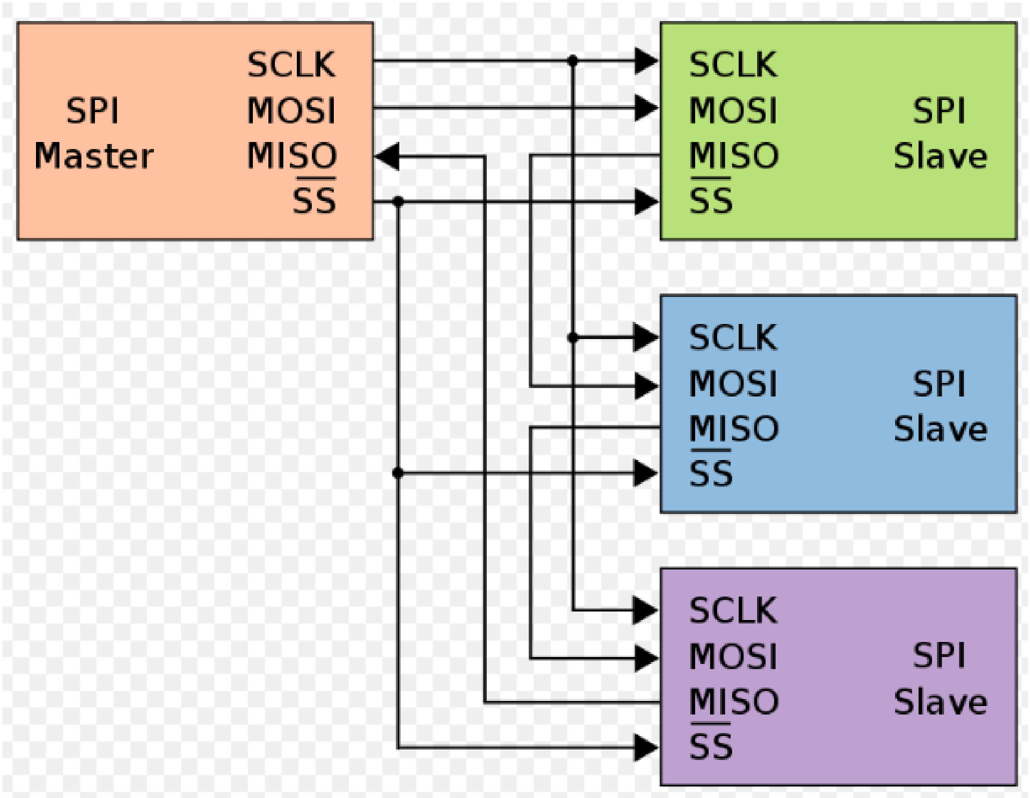
- Multiple SPI Slaves
  - One chip select for each slave
- Daisy Chaining
  - Inputs to Outputs
  - Chip Selects
- Dual or Quad SPI (or more lanes)
  - Implemented in high speed SPI Flash devices
  - Instead of one MISO, have N MISOs
  - N times bandwidth of traditional SPI
- 3 Wire (Microwire) SPI
  - Combined MISO/MOSI signal operates in half duplex

SCaLE 2019 - Introduction to SPI/SPIDEV

Multiple SPI Slaves

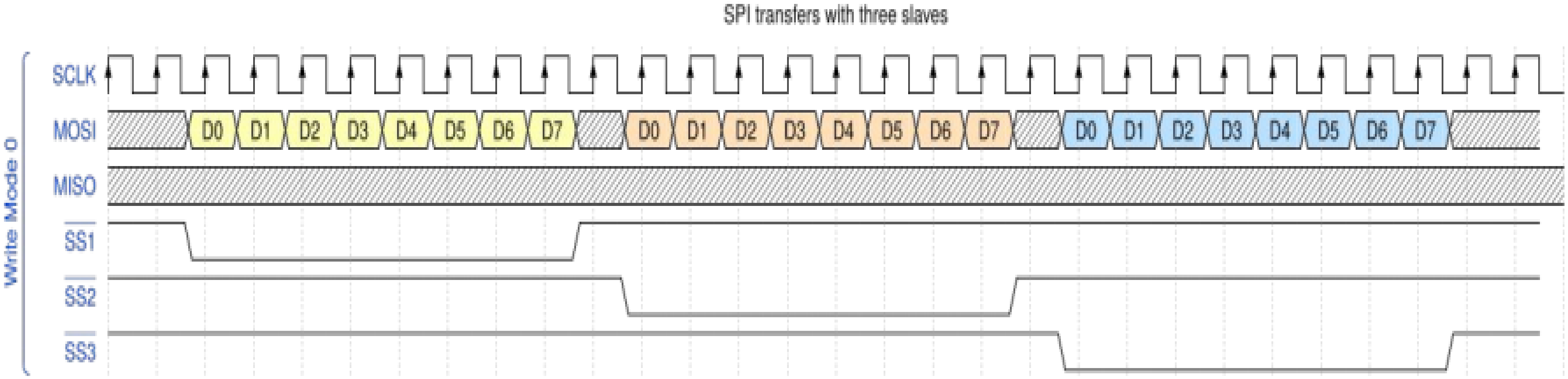


SPI Daisy Chain



SCaLE 2019 - Introduction to SPI/SPIDEV

SPI Mode Timing - Multiple Slaves



SCaLE 2019 - Introduction to SPI/SPIDEV

SPI hardware summary:

- Old Reliable Bus
- Still quite popular
- New variants are making it even more useful (QSPI, etc)

SCaLE 2019 - Introduction to SPI/SPIDEV

What is spidev?

- Generic pass-through SPI protocol driver
  - Works with SPI controller driver (as do other protocol drivers)
  - Alternative to protocol-specific SPI drivers
    - driver for SPI nor chip
    - driver for SPI GPIO chip

SCaLE 2019 - Introduction to SPI/SPIDEV

What does spidev do?

- Passes data between userspace and SPI controller
  - Collects buffers for tx/rx from userspace application
  - Hands off buffers to SPI controller driver
  - Returns back to userspace when transfer is complete

SCaLE 2019 - Introduction to SPI/SPIDEV

When should spidev be used?

- Prototyping in an environment that's not crash-prone; stray pointers in userspace won't normally bring down any Linux system
- Developing simple protocols used to talk to microcontrollers acting as SPI slaves, which you may need to change quite often

https://www.kernel.org/doc/Documentation/spi/spidev

SCaLE 2019 - Introduction to SPI/SPIDEV

When should spidev NOT be used?

- Of course there are drivers that can never be written in userspace, because they need to access kernel interfaces (such as IRQ handlers or other layers of the driver stack) that are not accessible to userspace

https://www.kernel.org/doc/Documentation/spi/spidev

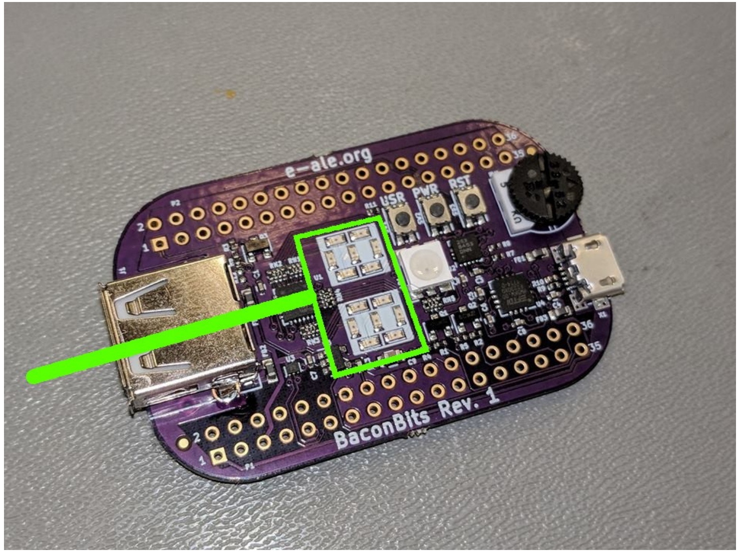
SCaLE 2019 - Introduction to SPI/SPIDEV

<u>Lab section</u>

SPIDEV

with

BaconBits LED controller



SCaLE 2019 - Introduction to SPI/SPIDEV

Lab #0

Prerequisite: install python spidev module

- pip3 install wheel-*.tar.gz
- pip3 install spidev-*.tar.gz

SCaLE 2019 - Introduction to SPI/SPIDEV

Lab #1

Goal: prove hardware is working

- run pre-built **`led_test`** app that flashes pattern on both LED groups

SCaLE 2019 - Introduction to SPI/SPIDEV

Lab #2

Goal: write your own software to make hardware do *something*

- create python tool for writing a value to LED gpios
  - collect hardware-level SPI info
  - fill it into python script template **template-write_value.py**

SCaLE 2019 - Introduction to SPI/SPIDEV

Lab #2

Answer:

- bus = 2
- device = 1
- max_speed_hz <= 10MHz
- mode = 0 or 3
- cmd = 0x40 $GROUP $VALUE
    - GROUP is 0x0 or 0x1 to select group of LEDs
    - VALUE is single byte specifying pattern for selected LEDs

SCaLE 2019 - Introduction to SPI/SPIDEV

Lab #3

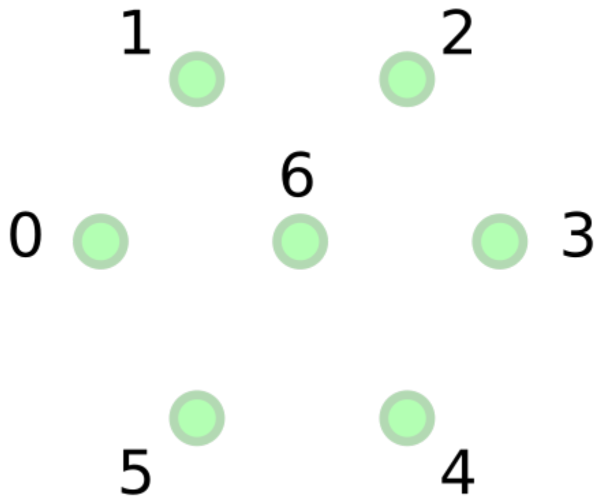
Goal: write your own tool for exploring hardware

- update python tool to set each bit in turn
  - record which bit controls which LED

SCaLE 2019 - Introduction to SPI/SPIDEV

Lab #3

Answer:



SCaLE 2019 - Introduction to SPI/SPIDEV

Lab #4

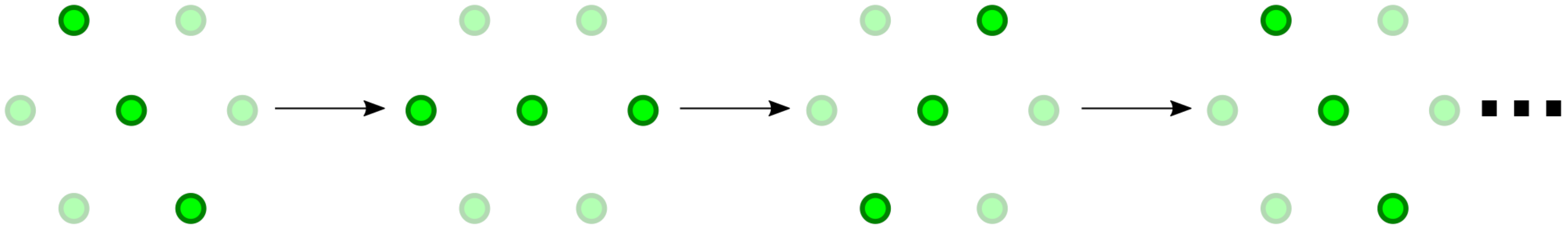
Goal: write software that actually does something useful with hardware

- use recorded bit-to-LED mapping to create spinner test pattern
- bonus: spin sets of LEDs in opposite directions

SCaLE 2019 - Introduction to SPI/SPIDEV

Lab #4

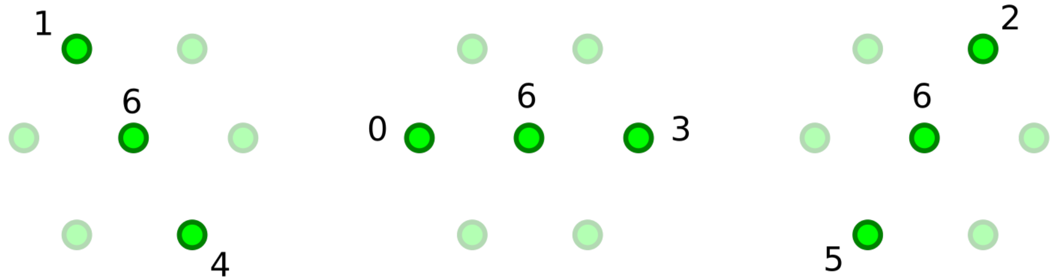
Spinner pattern



SCaLE 2019 - Introduction to SPI/SPIDEV

Lab #4

Answer:



1,6,4 = 0x02 | 0x40 | 0x10 = 0x52 ==> inverted = 0x52 ^ 0xff = 0xad

0,6,3 = 0x01 | 0x40 | 0x08 = 0x49 ==> inverted = 0x49 ^ 0xff = 0xb6

2,6,5 = 0x04 | 0x40 | 0x20 = 0x64 ==> inverted = 0x64 ^ 0xff = 0x9b