EXPLIOT
www.expliot.io

Introduction To Hardware

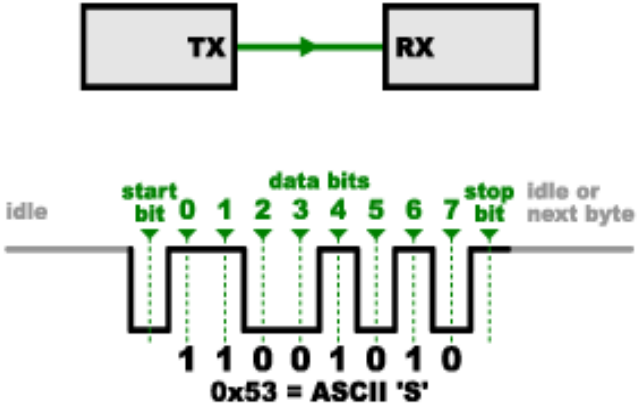
SPI

Introduction to SPI

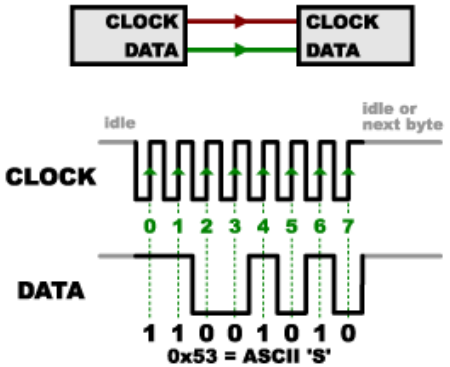
Introduction

- Serial Peripheral Interface
- Master slave communication
- Synchronous serial communication
- The receive hardware can be a simple shift register
- High speed communication
- It supports multiple peripherals
- No ACK protocol
- Most frequently used in embedded devices

Need For Synchronous Communication



Asynchronous Communication



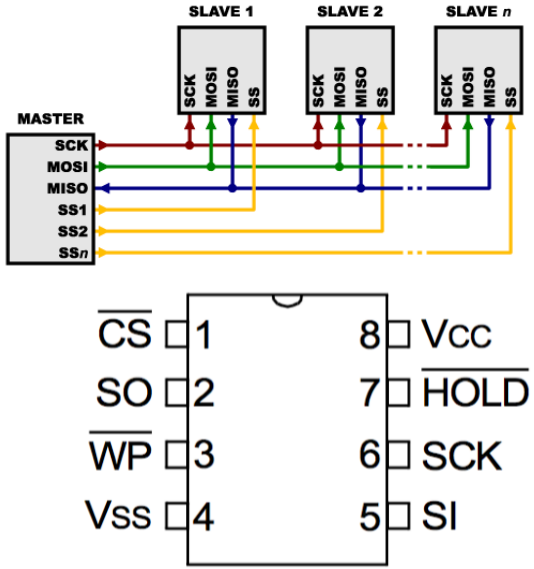
Synchronous Communication

Source : https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi/all

SPI Protocol

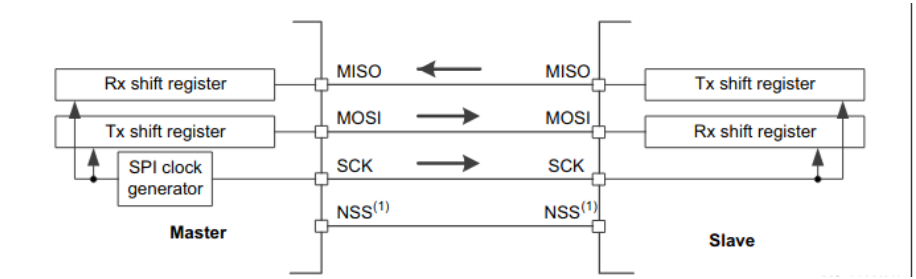
- It stands for serial peripheral interface.
- Developed by Motorola
- Four wire Synchronous Protocol

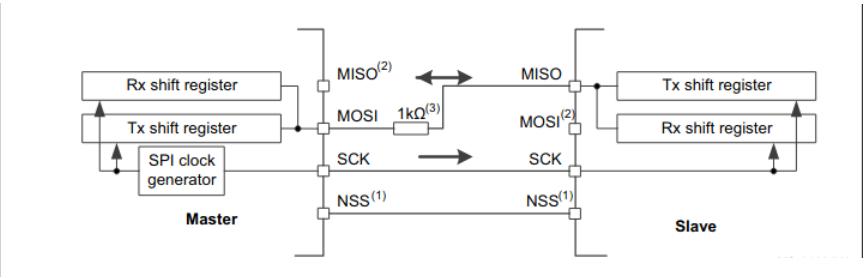
- MOSI (Master Output, Slave Input),
- MISO (Master Input, Slave Output)
- SS/CS (Slave Select/Chip Select)
- SCLK (Serial Clock).
- It's faster than asynchronous serial.
- CS line decides which device to communicate with.

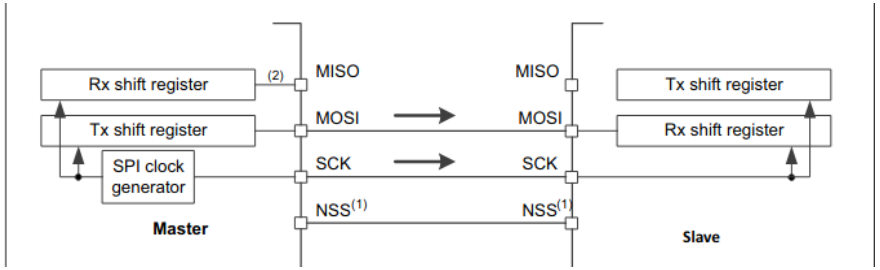


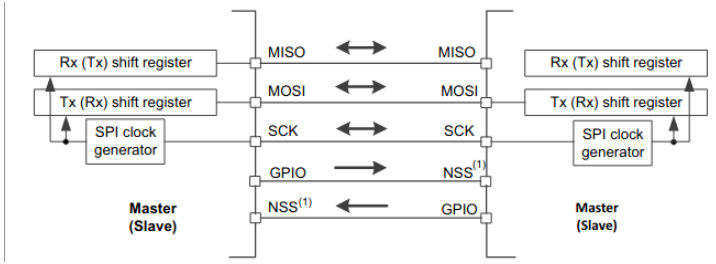
Source - Wikipedia

SPI Master- Slave Communication



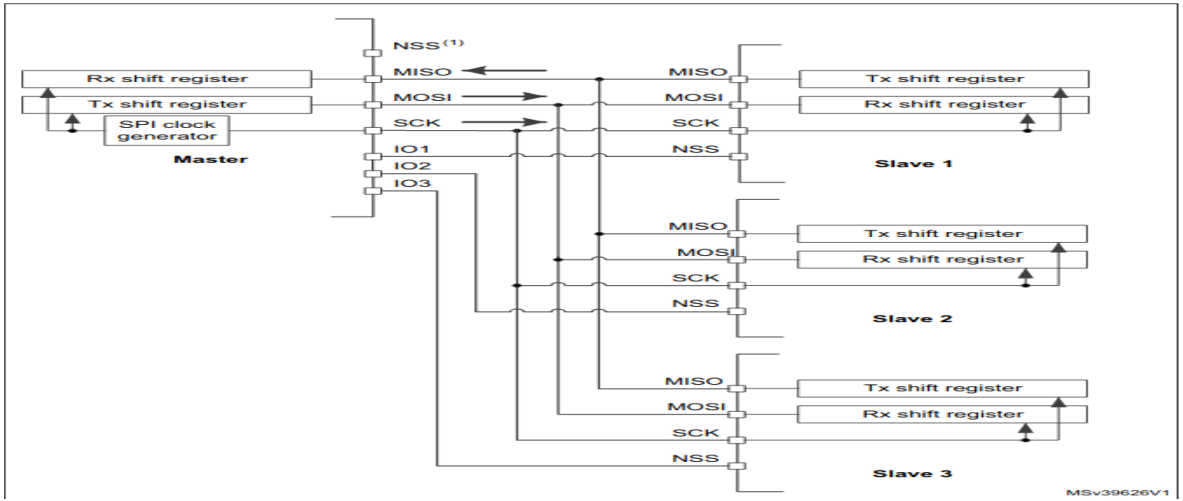






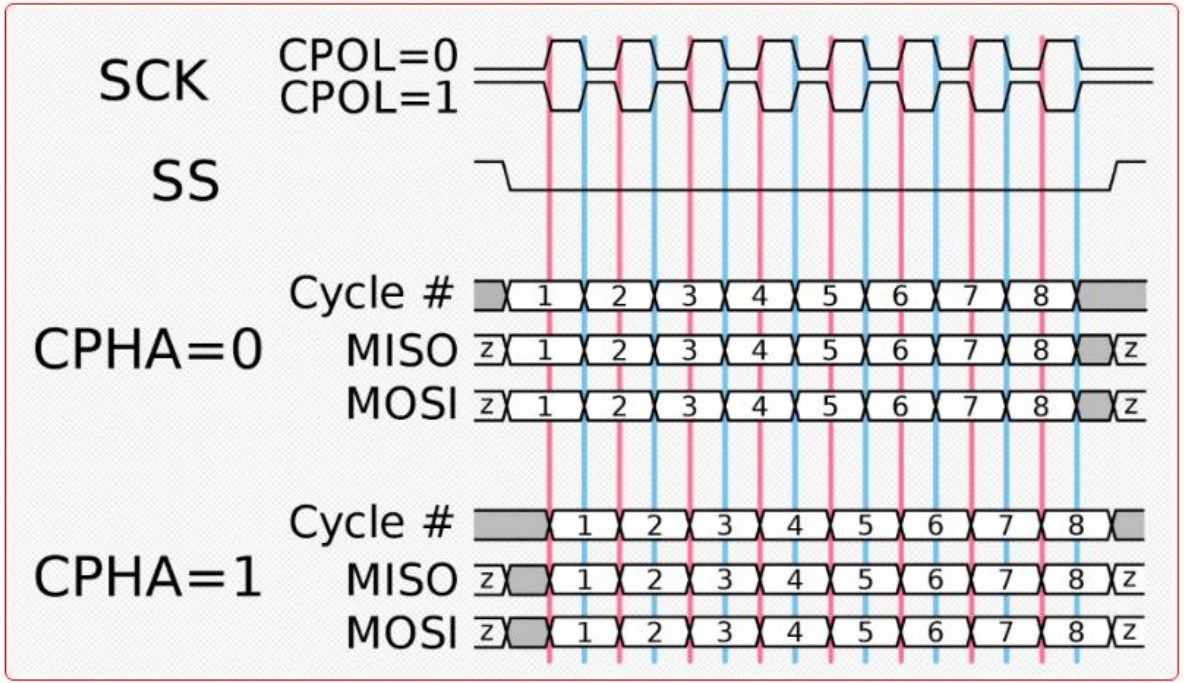
Source: STM32L4R29 Reference Manual

SPI Master- Slave Communication



Source: STM32L4R29 Reference Manual

SPI Protocol



Source - Wikipedia

SPI Protocol

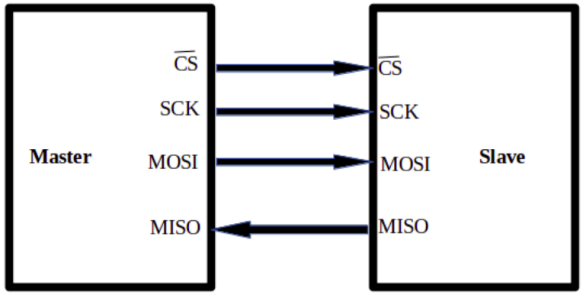
- Full-duplex
- Master slave architecture
- Single master multiple slaves
- Modes of SPI - Mode 0, Mode 1, Mode 2, Mode 3

SPI Modes	Clock Polarity (CPOL/CKP)	Clock Phase (CPHA/CKE)
Mode 0	0	0
Mode 1	0	1
Mode 2	1	0
Mode 3	1	1

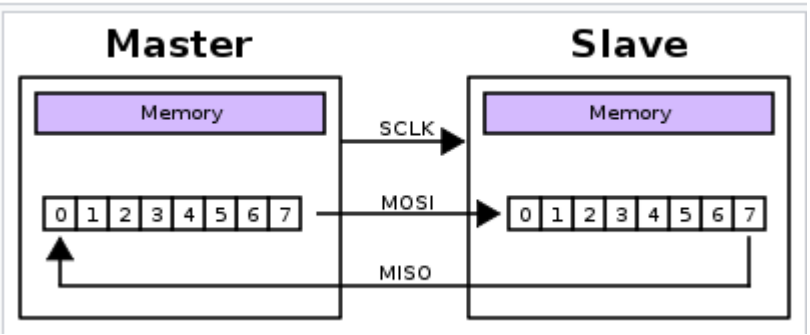
Source - Wikipedia

SPI Protocol

- Master selects the slave device
- Master sends the clock signal
- Master sends the data/command on the MOSI line
- Slave sends the data/response on the MISO line



Source - Wikipedia



SPI Protocol

- Master selects the slave device
- Master sends the clock signal
- Master sends the data/command on the MOSI line
- Slave sends the data/response on the MISO line

Source - Wikipedia

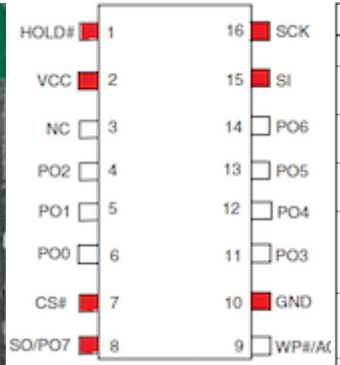
Typical SPI Chips

- RF transceiver
  - Nrf24l01
  - CC 1100
- MEMS sensors
  - LIS3DH12
  - IIS3WB
- Memory
  - W25Q128
  - CY15B104Q

SPI - Possible Attacks

- Signal sniffing
- Signal tampering
- Data extraction
- Data manipulation

Memory Extracted



Signal Name	I/O	Description
SO (Signal Data Output)	Output	Transfers data serially out of the device on the falling edge of SCK.
PO[7–0] (Parallel Data Input/Output)	Input/Output	Transfers parallel data into the device on the rising edge of SCK or out of the device on the falling edge of SCK.
SI (Serial Data Input)	Input	Transfers data serially into the device. Device latches commands, addresses, and program data on SI on the rising edge of SCK.
SCK (Serial Clock)	Input	Provides serial interface timing. Latches commands, addresses, and data on SI on rising edge of SCK. Triggers output on SO after the falling edge of SCK.
CS# (Chip Select)	Input	Places device in active power mode when driven low. Deselects device and places SO at high impedance when high. After power-up, device requires a falling edge on CS# before any command is written. Device is in standby mode when a program, erase, or Write Status Register operation is not in progress.
HOLD# (Hold)	Input	Pauses any serial communication with the device without deselecting it. When driven low, SO is at high impedance, and all input at SI and SCK are ignored. Requires that CS# also be driven low.
WP#/ACC (Write Protect/Accelerated Programming)	Input	When driven low, prevents any program or erase command from altering the data in the protected memory area specified by Status Register bits (BP bits). If the system asserts V_{HH} on this pin, accelerated programming operation is provided.
V_{CC}	Input	Supply Voltage
GND	Input	Ground

Architectural Advantages

- Single power supply operation
  - Full voltage range: 2.7V to 3.6V read and program operations
- Memory Architecture
  - 128Mb uniform 256 KB sector product
  - 128Mb uniform 64 KB sector product

Software Features

- SPI Bus Compatible Serial Interface

- Process Technology
  - Manufactured on 0.09 µm MirrorBit® process technology
- Package Option
  - Industry Standard Pinouts
  - 16-pin SO package (300 mils)
  - 8-Contact WSON Package (6 x 8 mm)

Source: https://www.nsideattacklogic.de/en/dumping-spi-flash-memory-of-embedded-devices-2/

Memory Extracted

```
pi@icbm:~ $ sudo flashrom -p linux_spi:dev=/dev/spidev0.0,spispeed=8000 -c S25
FL128P......0 -r S25FL128P-dump.bin

flashrom p1.0-65-ga9a03cc-dirty on Linux 4.9.35-v7+ (armv7l)
flashrom is free software, get the source code at https://flashrom.org

Using clock_gettime for delay loops (clk_id: 1, resolution: 1ns).
Found Spansion flash chip "S25FL128P......0" (16384 kB, SPI) on linux_spi.
===
This flash part has status UNTESTED for operations: PROBE READ ERASE WRITE
The test status of this chip may have been updated in the latest development
version of flashrom. If you are running the latest development version,
please email a report to flashrom@flashrom.org if any of the above operations
work correctly for you with this flash chip. Please include the flashrom log
file for all operations you tested (see the man page for details), and mention
which mainboard or programmer you tested in the subject line.
Thanks for your help!
Reading flash... done.
```

Source: https://www.nsideattacklogic.de/en/dumping-spi-flash-memory-of-embedded-devices-2/

Memory Extracted

```
pi@icbm:~ $ sudo flashrom -p linux_spi:dev=/dev/spidev0.0,spispeed=8000

flashrom p1.0-65-ga9a03cc-dirty on Linux 4.9.35-v7+ (armv7l)
flashrom is free software, get the source code at https://flashrom.org

Using clock_gettime for delay loops (clk_id: 1, resolution: 1ns).
Found Spansion flash chip "S25FL127S-64kB" (16384 kB, SPI) on linux_spi.
Found Spansion flash chip "S25FL127S-256kB" (16384 kB, SPI) on linux_spi.
Found Spansion flash chip "S25FL128P.....0" (16384 kB, SPI) on linux_spi.
Found Spansion flash chip "S25FL128P.....1" (16384 kB, SPI) on linux_spi.
Found Spansion flash chip "S25FL128S.....0" (16384 kB, SPI) on linux_spi.
Found Spansion flash chip "S25FL128S.....1" (16384 kB, SPI) on linux_spi.
Found Spansion flash chip "S25FL129P.....0" (16384 kB, SPI) on linux_spi.
Found Spansion flash chip "S25FL129P.....1" (16384 kB, SPI) on linux_spi.
Multiple flash chip definitions match the detected chip(s): "S25FL127S-64kB",
"S25FL127S-256kB", "S25FL128P.....0", "S25FL128P.....1", "S25FL128S.....0",
 "S25FL128S.....1", "S25FL129P.....0", "S25FL129P.....1"
Please specify which chip definition to use with the -c <chipname> option.
```

Source: https://www.nsideattacklogic.de/en/dumping-spi-flash-memory-of-embedded-devices-2/

Memory Extracted

```
pi@icbm:~ $ sha1sum S25FL128*
93f3cd051264023d84e9e44fab8794de9dee6933  S25FL128P-dump_02.bin
93f3cd051264023d84e9e44fab8794de9dee6933  S25FL128P-dump_03.bin
93f3cd051264023d84e9e44fab8794de9dee6933  S25FL128P-dump.bin
```

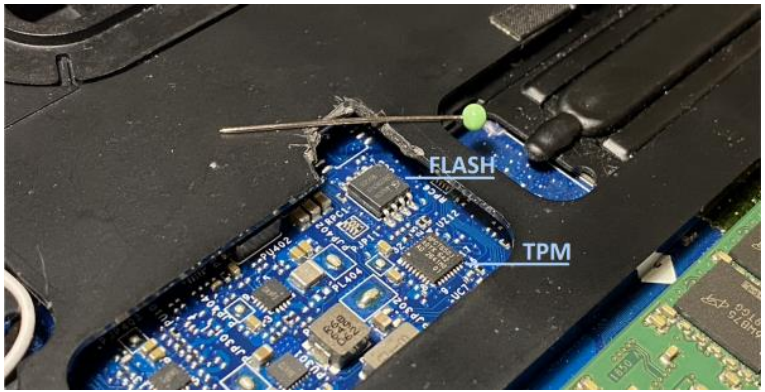
Source: https://www.nsideattacklogic.de/en/dumping-spi-flash-memory-of-embedded-devices-2/

Memory Extracted

```
pi@icbm:~ $ strings -20 S25FL128P-dump.bin | head -16
U-Boot 1.2.0-g73dbfe77 (May 27 2011 - 10:28:29)
PSPU-Boot 1.0.16.22-H2.8.3
Warning: the sector size of two banks of flash are different.
Unsupported operation '%s'
Unknown/Unsupported operator '%s'
eval      - return addition/subraction
[*]value1 <op> [*]value2
Error: Sector amount %d exceding FLASH maximum sectors %d
Error: Missing or unknown FLASH type
Size: %ld MB in %d Sectors
Sector Start Addresses:
Serial Flash  Read JEDEC Failed
Error: Failed to read status register
Error: Serial FLASH busy
- no sectors to erase
 sf_erase_sector failed for sector = %d address = 0x%x
```

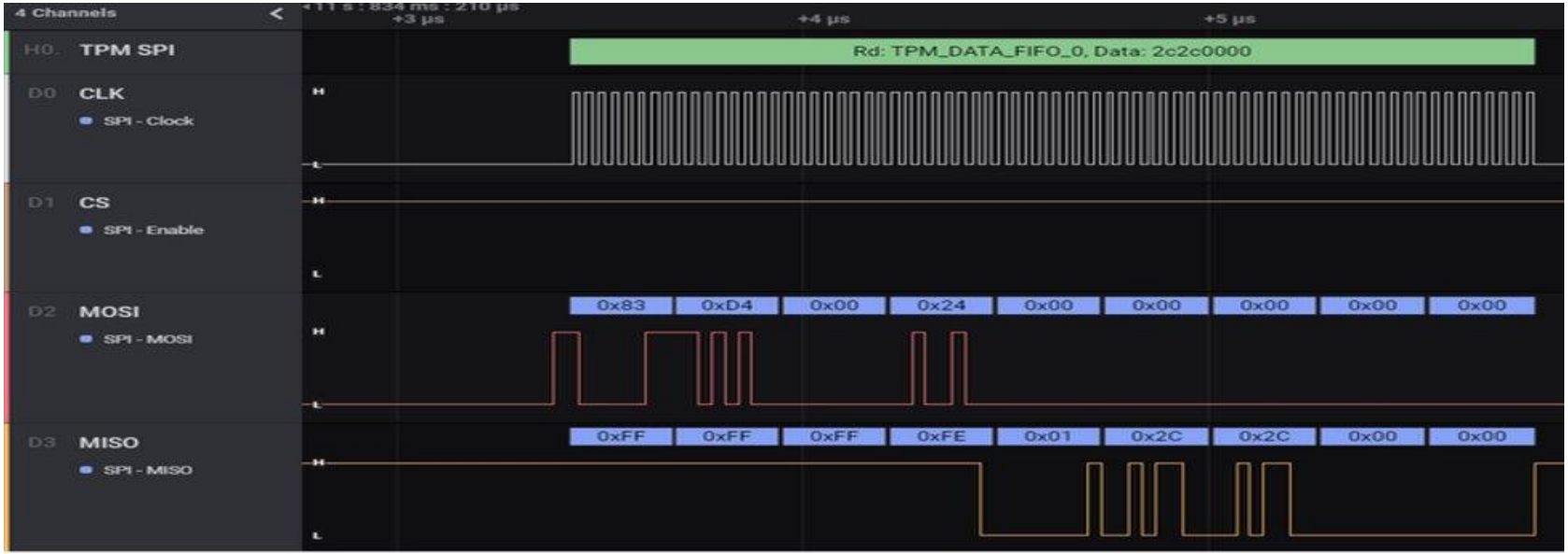
Source: https://www.nsideattacklogic.de/en/dumping-spi-flash-memory-of-embedded-devices-2/

Attack on TPM



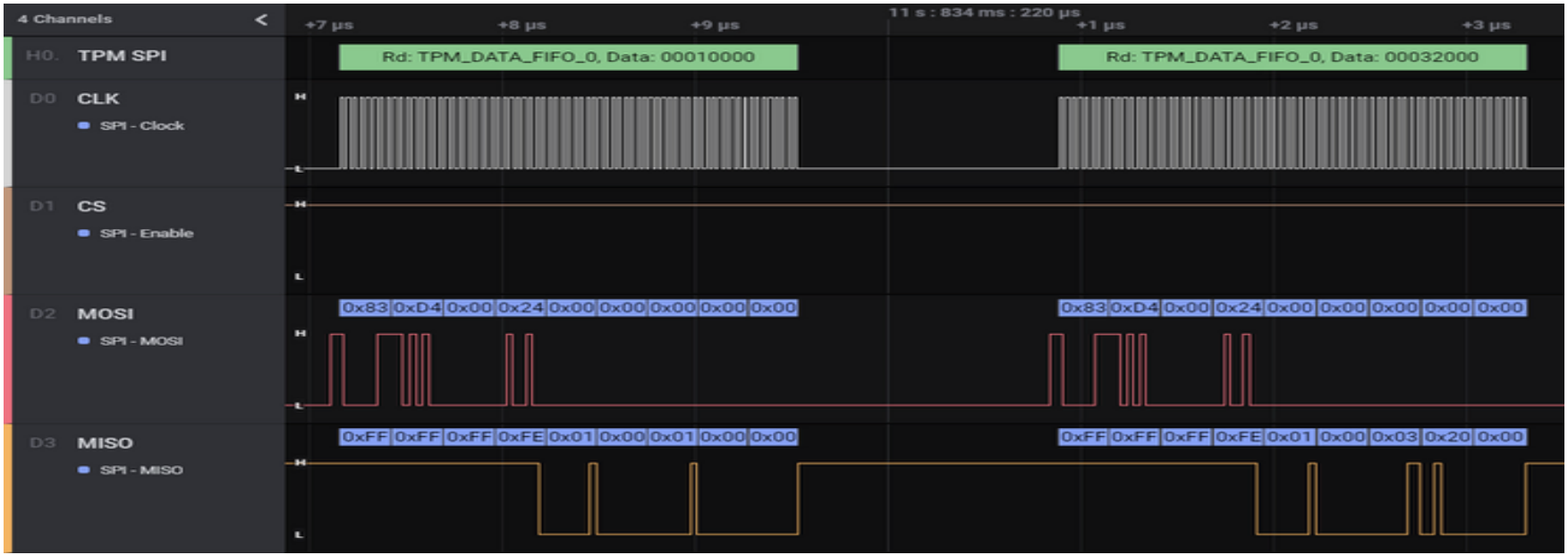
Source: https://labs.f-secure.com/blog/sniff-there-leaks-my-bitlocker-key/

TPM Sniffed



Source: https://www.cryptic.red/post/sniffing-tpm-keys-on-the-spi-bus

TPM Sniffed



Source: https://www.cryptic.red/post/sniffing-tpm-keys-on-the-spi-bus

TPM Sniffed

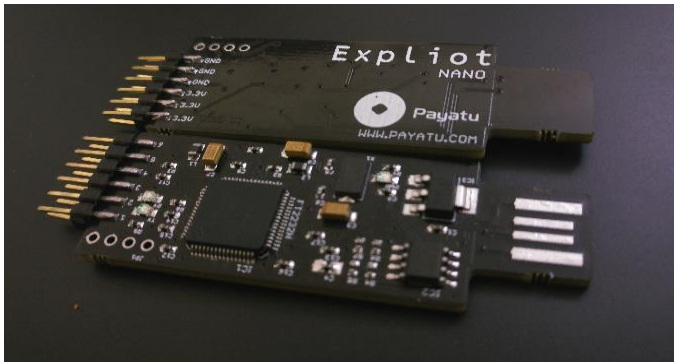
Source: https://www.cryptic.red/post/sniffing-tpm-keys-on-the-spi-bus

SPI - Tools/Framework

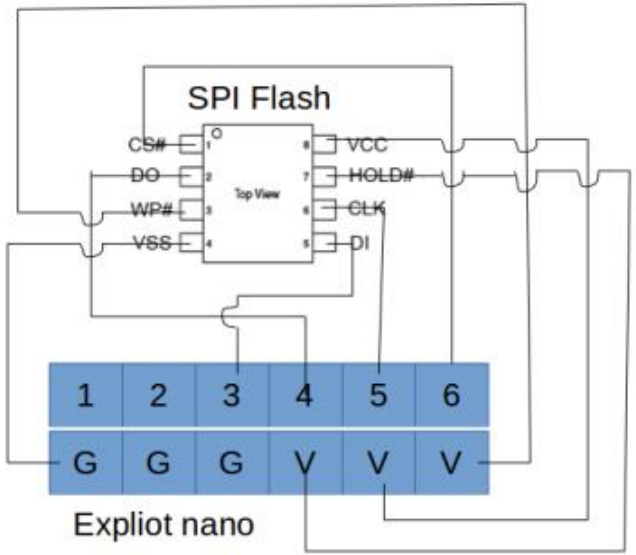
- EXPLIoT Nano - https://expliot.io/products/expliot-nano
- Bus Pirate - http://dangerousprototypes.com/docs/Bus_Pirate
- Shikra - https://int3.cc/products/the-shikra
- CH341A - https://www.onetransistor.eu/2017/08/ch341a-mini-programmer-schematic.html
- EXPLIoT Framework - https://gitlab.com/expliot_framework/expliot
- Flashrom - https://github.com/flashrom/flashrom
- Pyspiflash - https://github.com/eblot/pyspiflash
- Logic Analyzer
- RaspberryPi or Beaglebone can also be used.

EXPLIoT Nano

- Expliot-NANO is a compact hacker friendly multi-purpose, multi-protocol hardware tool mainly used to debug and program microcontrollers/processors and flash chips
- Expliot-NANO can be configured to support hardware protocols including, UART, I2C, SPI, ARM SWD and JTAG.



Interfacing



The End