Slides and materials at:

https://github.com/thedukezip/BSidesBoston2016

Windows and OS X users:

Kali VM or boot off USB

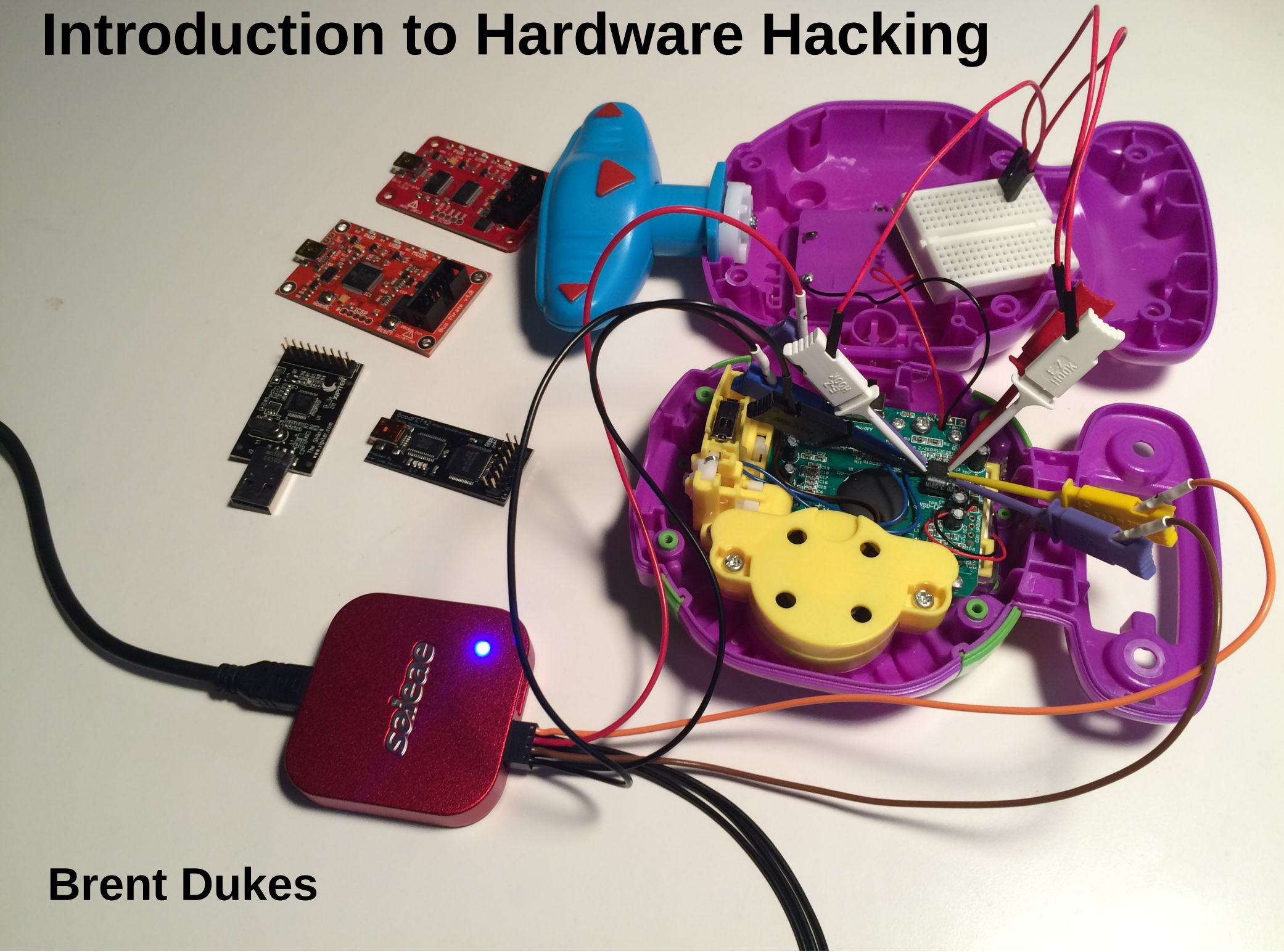
Linux users:

You're probably all set!

VirtualBox Guest Additions:

```
apt-get update
apt-get install -y virtualbox-guest-x11
reboot
```

Introduction to Hardware Hacking



Brent Dukes

Brent Dukes

@TheDukeZip

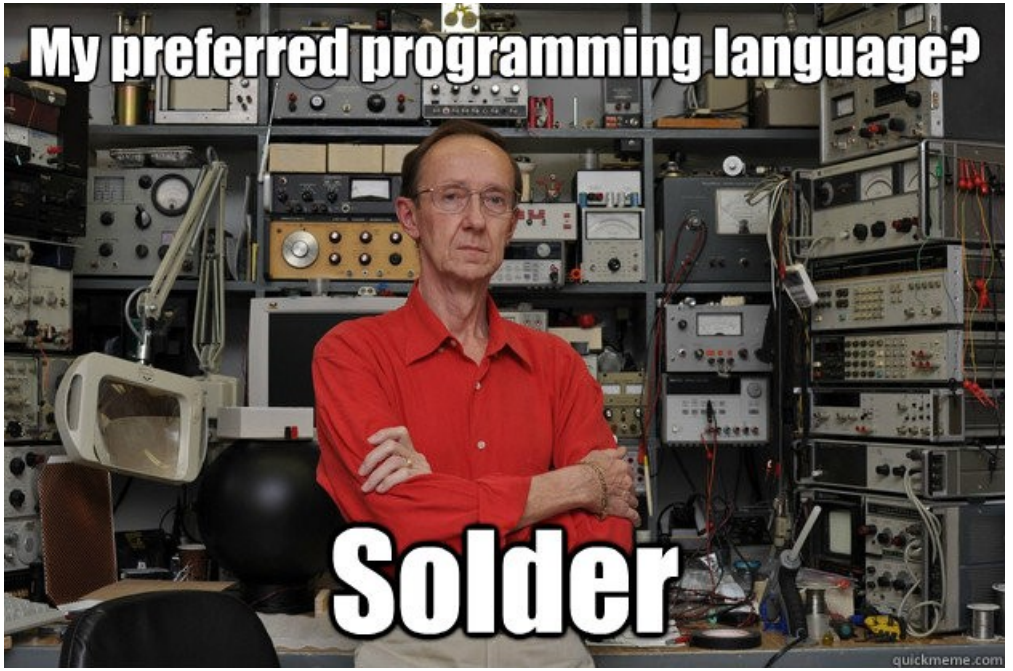
Systems Engineer – 12 yrs

Radio systems design end to end

Avid CTFer

What This Class **ISN'T**

- Learn how to solder
- Design sophisticated circuits
- Build LED blinking things (well...)



What This Class **IS**

What This Class **IS**

Reverse engineering embedded systems

Agenda

Building blocks of an embedded system

Components – ID and packaging

Circuit board design and features

Communication busses

Lab 1

LUNCH

JTAG

Lab 2

Radio modules

Lab 3

Lab 4

Labs

Lab 1

Extract data from flash storage

Lab 2

Extract firmware from microcontroller

Dissassemble and examine firmware (in ASM)

Lab 3

Extract data from flash storage

Lab 4

Sniff communication bus and decode information transmitted

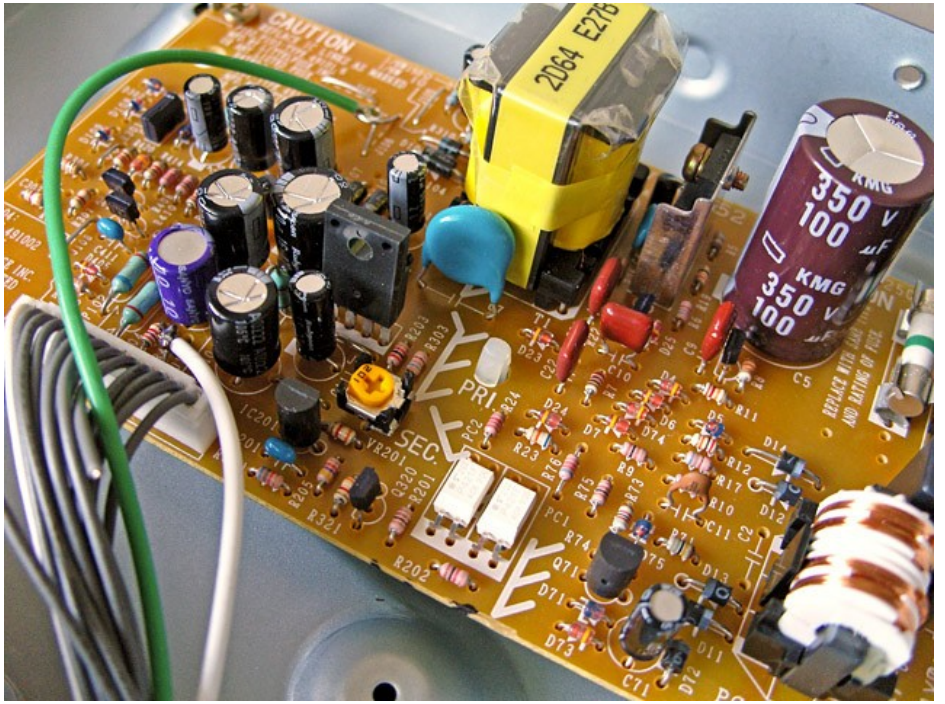

GoodFET42
MSP-EXP430G2
LaunchPad
TEXAS INSTRUMENTS

Safety!

Use caution when disassembling electrical equipment

- ✔ Battery Powered
- ✔ USB Powered
- ✔ AC to DC Adapter powered

- ✗ Plugs directly into wall socket

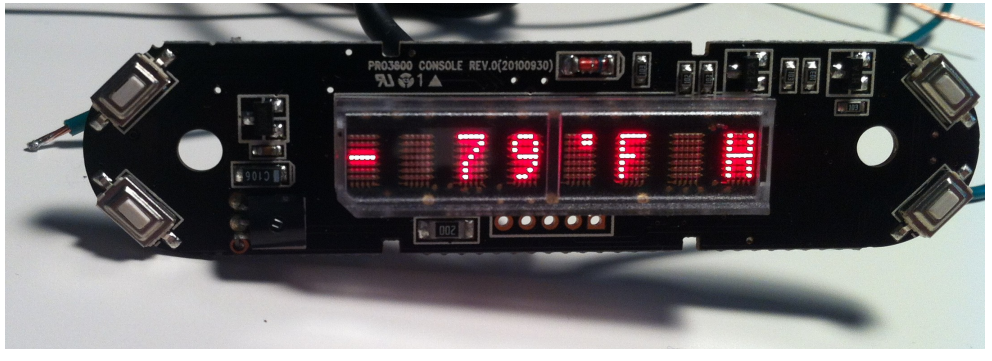


Why Hardware Hacking?

- Bypass certificate pinning for MITM
- Gain terminal access to device
- Reverse engineer protocols used
- Modify device for unintended operation

Why Hardware Hacking?

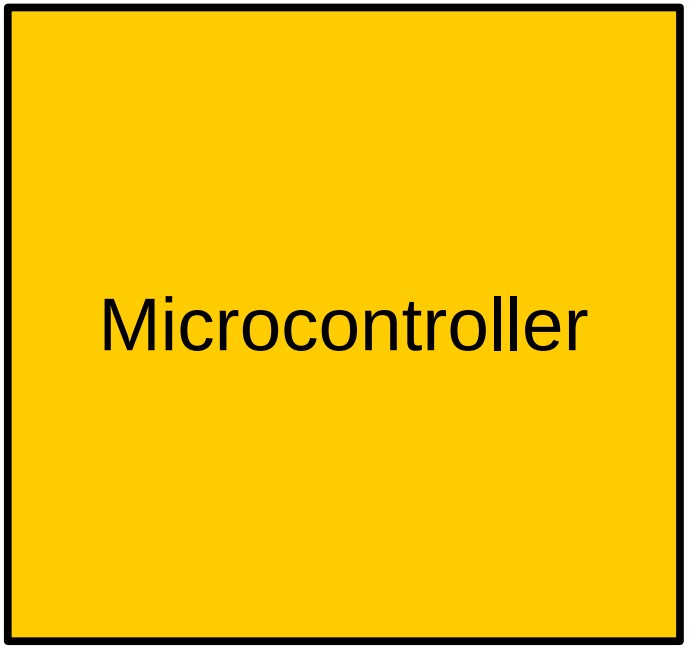
- Bypass certificate pinning for MITM
- Gain terminal access to device
- Reverse engineer protocols used
- Modify device for unintended operation



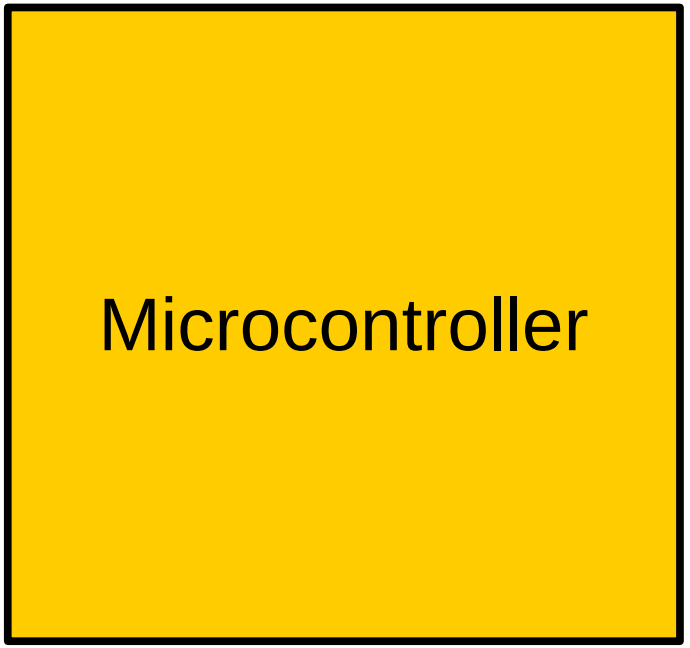
Embedded Systems

What is an embedded system?

Embedded System

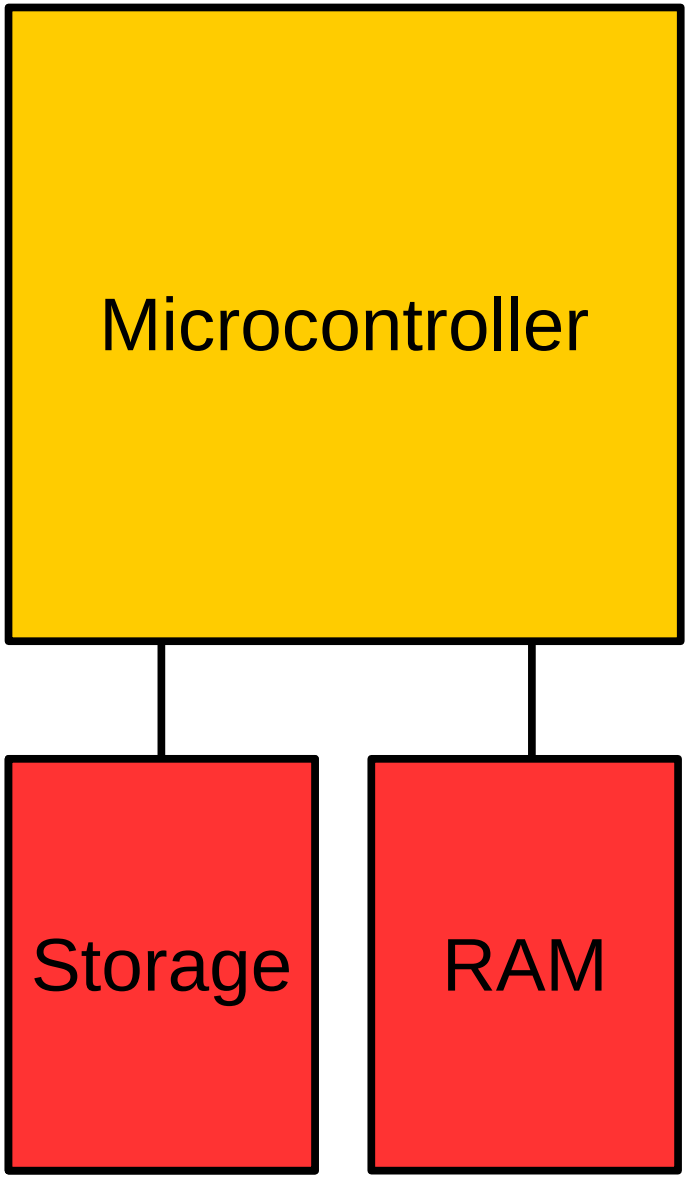


Embedded System

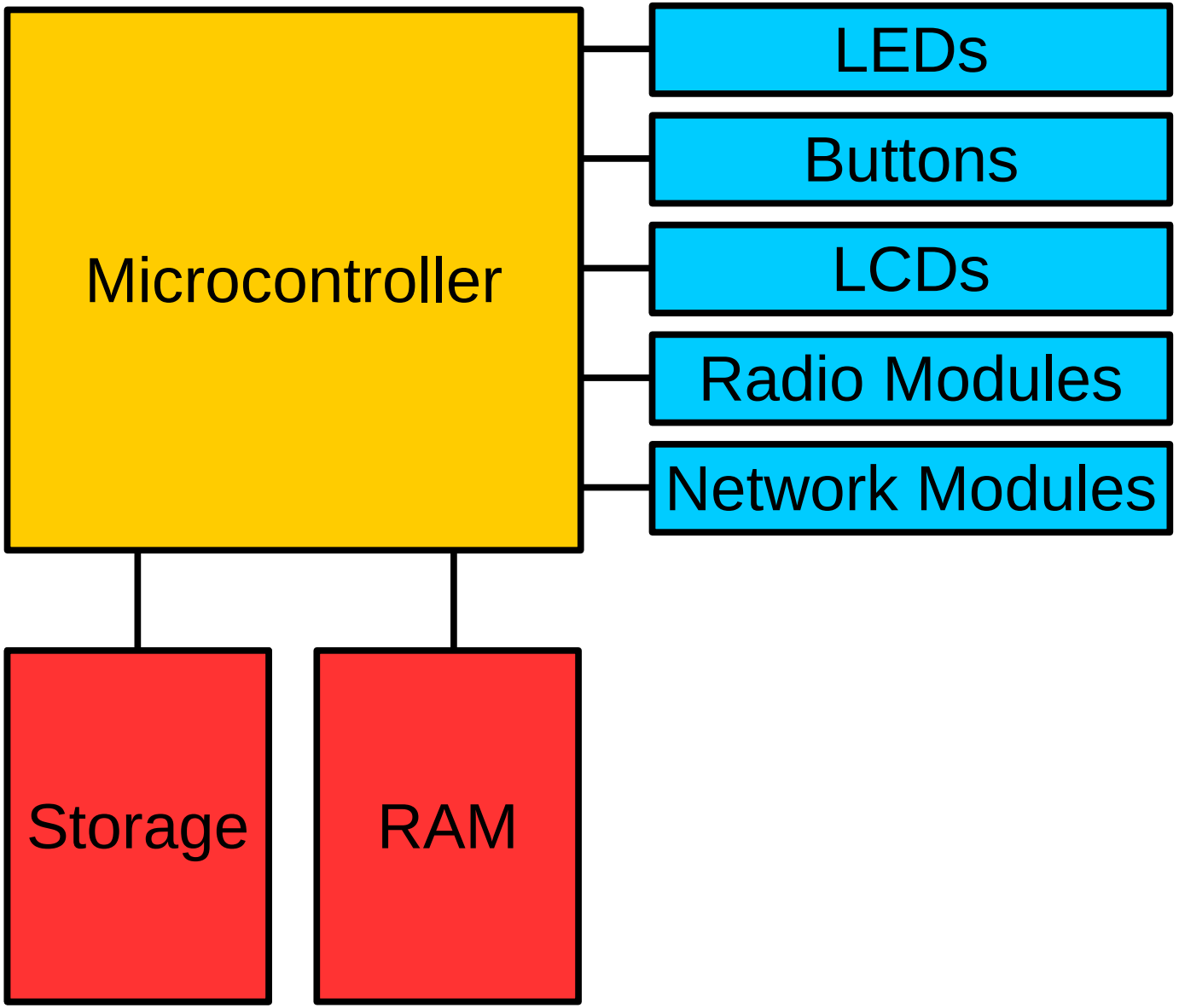


- Non-volatile storage
- RAM
- Processor
- Integrated Modules
  - Timers
  - I/O
  - A/D D/A
  - LCD controllers
  - Capacitive touch controllers
  - ….

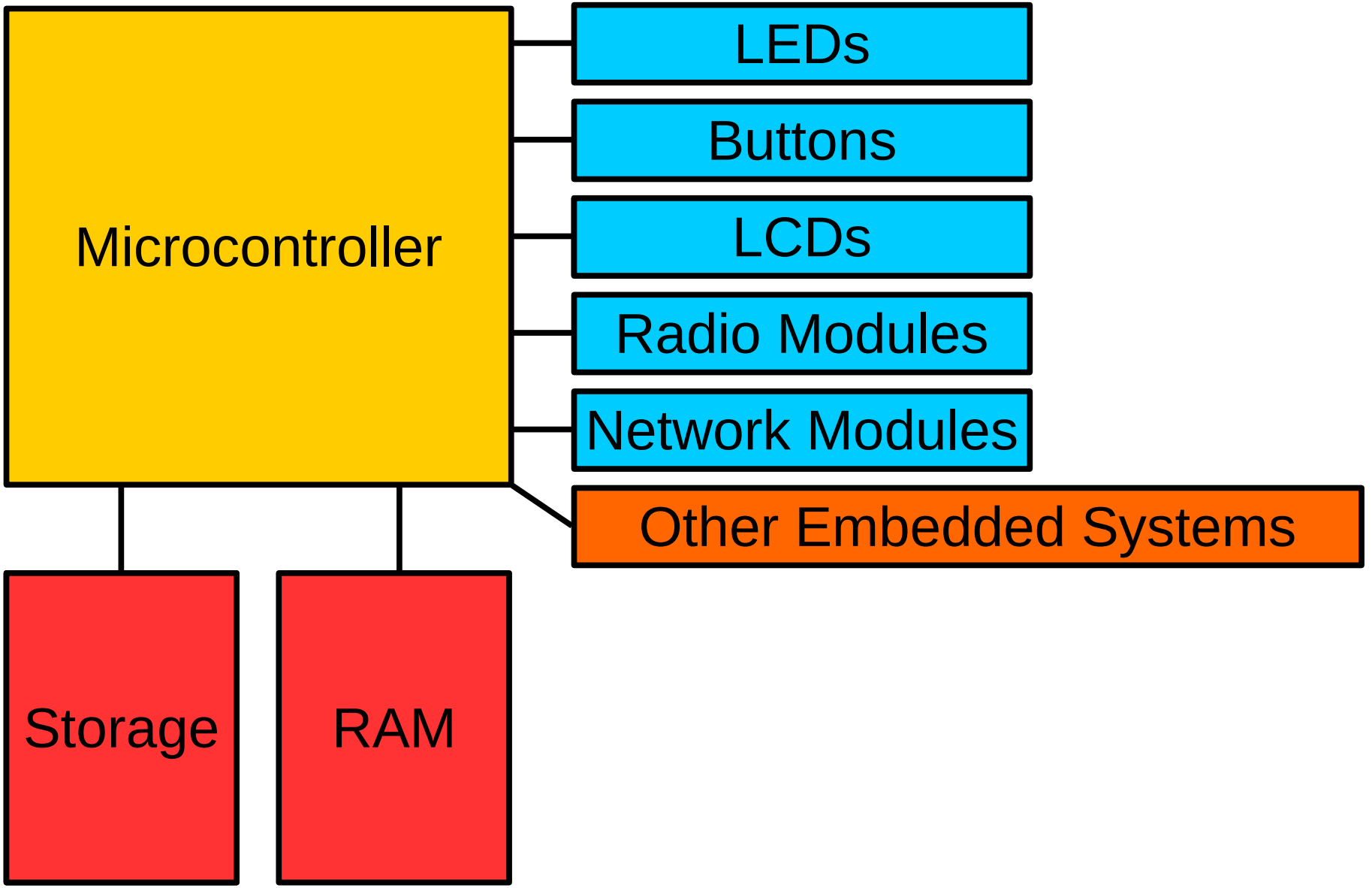
Embedded System



Embedded System



Embedded System



Component Identification


54CVOPK A
M430G2553

Datasheets

MIXED SIGNAL MICROCONTROLLER

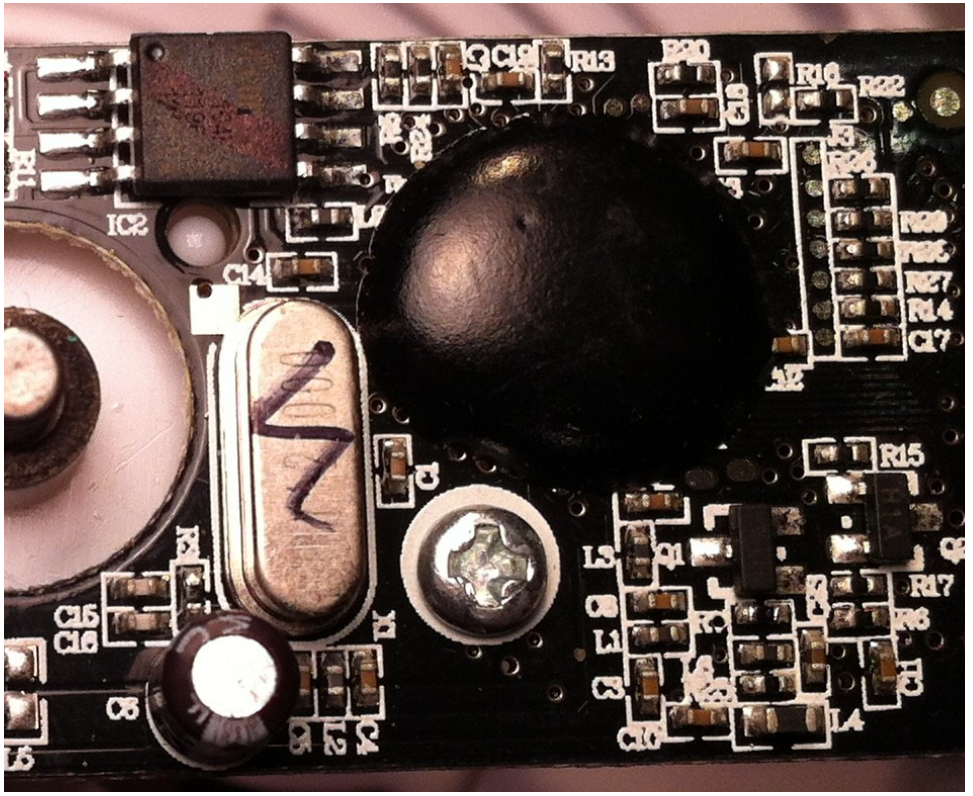
FEATURES

- Low Supply-Voltage Range: 1.8 V to 3.6 V
- Ultra-Low Power Consumption
  - Active Mode: 230 µA at 1 MHz, 2.2 V
  - Standby Mode: 0.5 µA
  - Off Mode (RAM Retention): 0.1 µA
- Five Power-Saving Modes
- Ultra-Fast Wake-Up From Standby Mode in Less Than 1 µs
- 16-Bit RISC Architecture, 62.5-ns Instruction Cycle Time
- Basic Clock Module Configurations
  - Internal Frequencies up to 16 MHz With Four Calibrated Frequency
  - Internal Very-Low-Power Low-Frequency (LF) Oscillator
  - 32-kHz Crystal
  - External Digital Clock Source
- Two 16-Bit Timer_A With Three
- Universal Serial Communication Interface (USCI)
  - Enhanced UART Supporting Auto Baudrate Detection (LIN)
  - IrDA Encoder and Decoder
  - Synchronous SPI
  - I^2C™
- On-Chip Comparator for Analog Signal Compare Function or Slope Analog-to-Digital (A/D) Conversion
- 10-Bit 200-ksps Analog-to-Digital (A/D) Converter With Internal Reference, Sample-and-Hold, and Autoscan (See Table 1)
- Brownout Detector
- Serial Onboard Programming, No External Programming Voltage Needed, Programmable Code Protection by Security Fuse
- On-Chip Emulation Logic With Spy-Bi-Wire

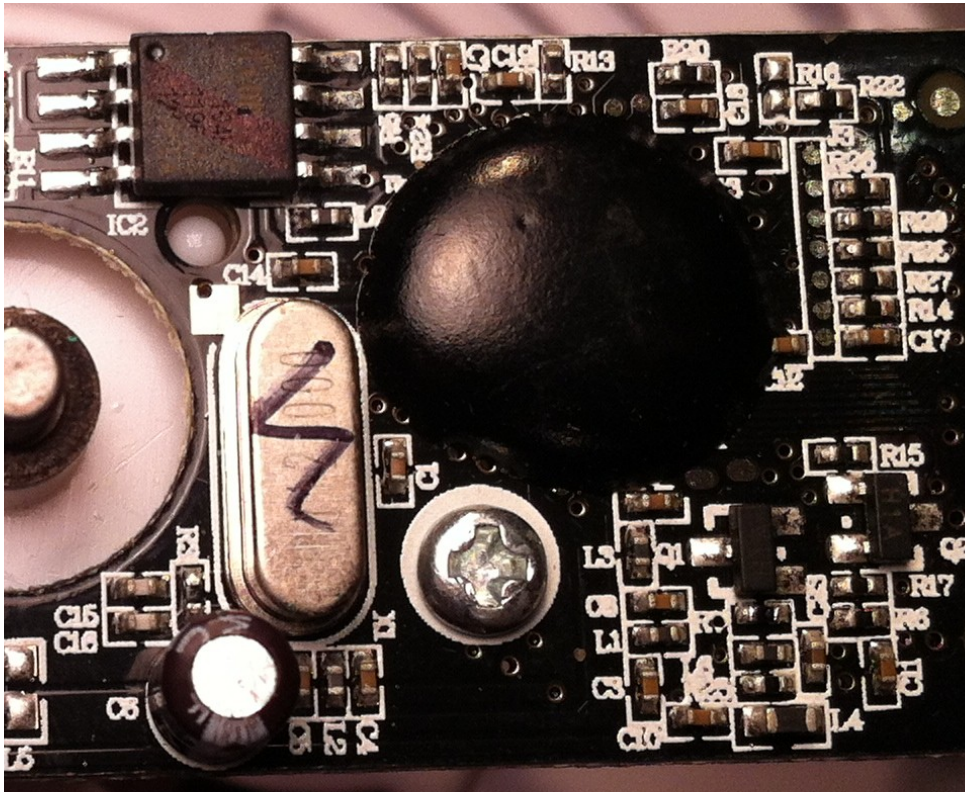
Component Identification

- Custom Part Numbers
  - Research manufacturer
  - Utilize programming port
  - Sniff communications
  - Reverse engineer pinout

Component Identification



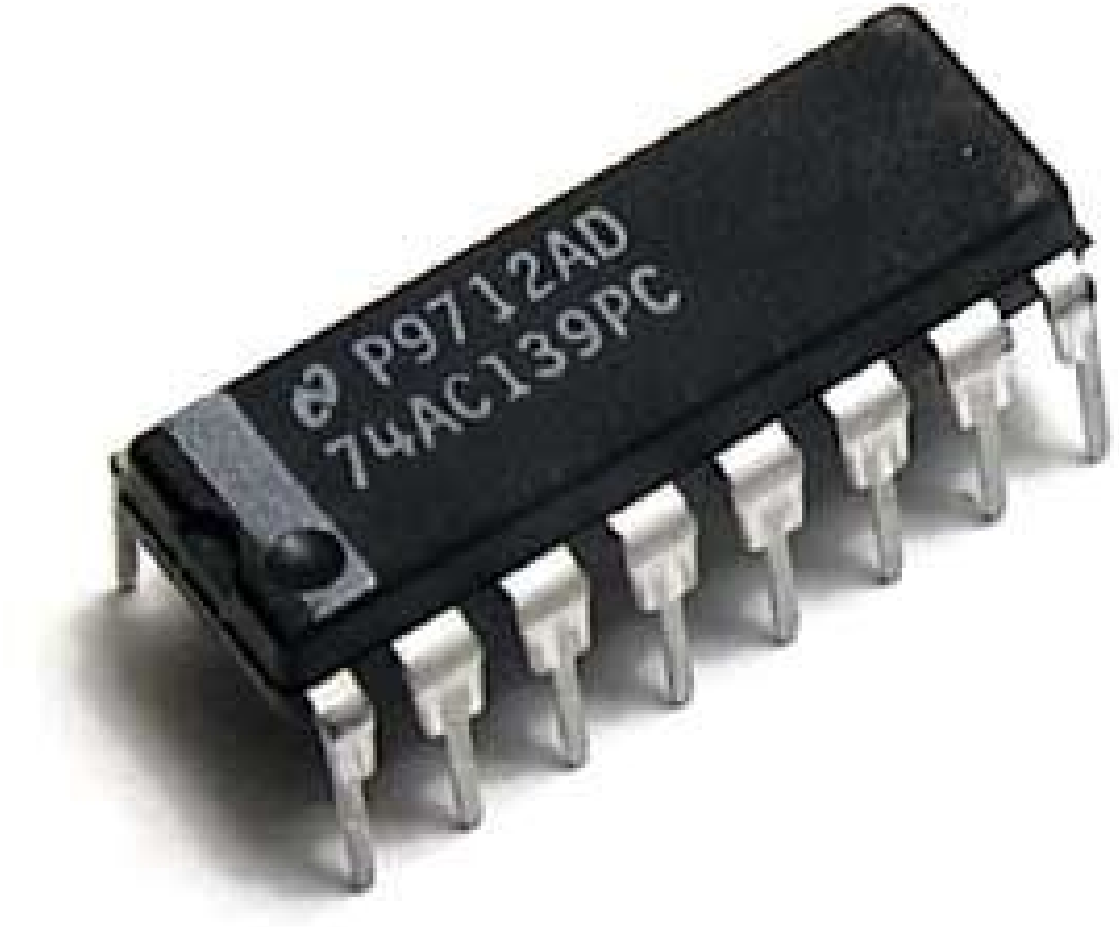
Component Identification



- Epoxy
  - Utilize programming port
  - Sniff communications
  - Reverse engineer pinout

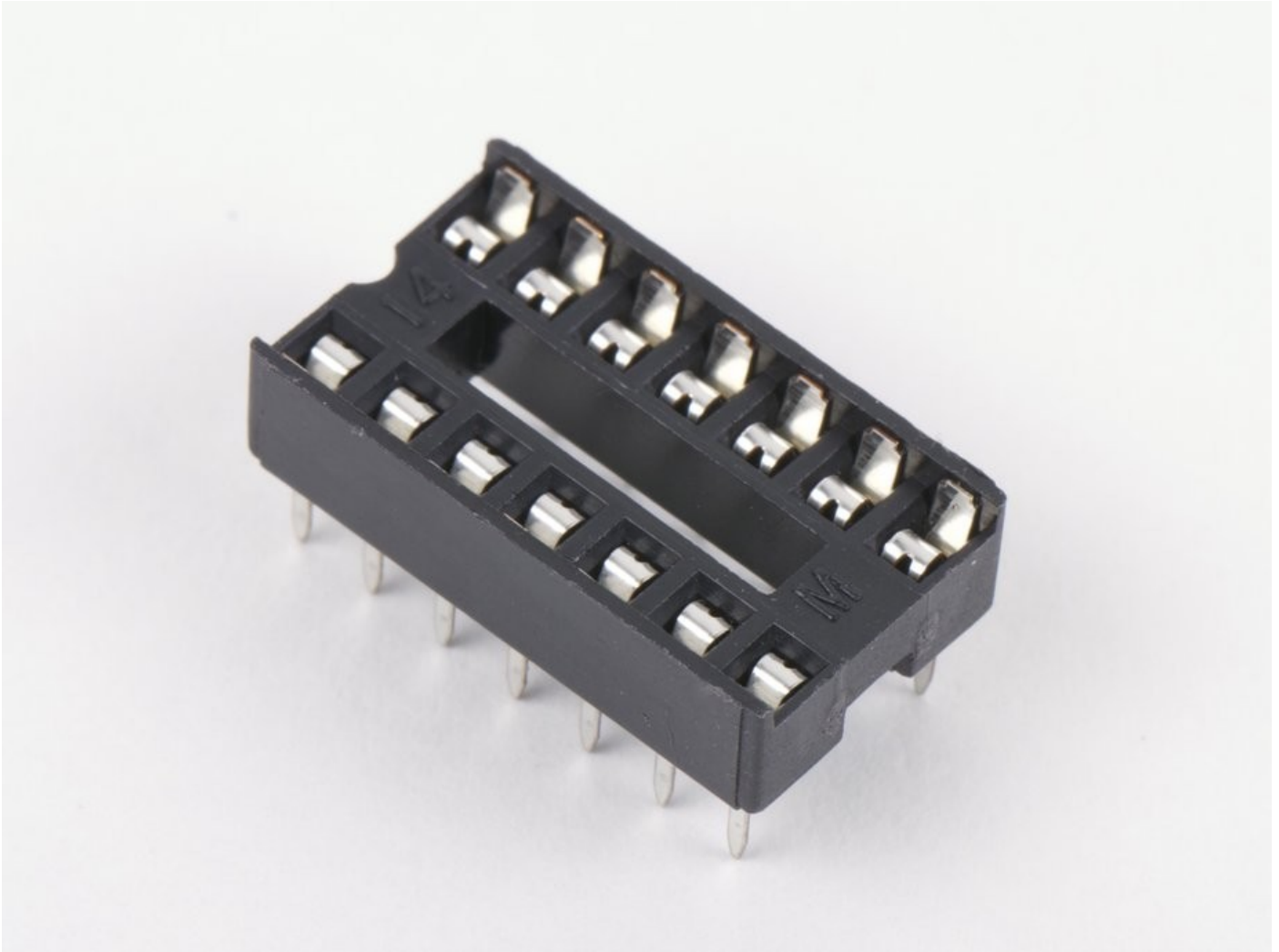
IC Packaging

DIP (Dual Inline Package)



IC Packaging

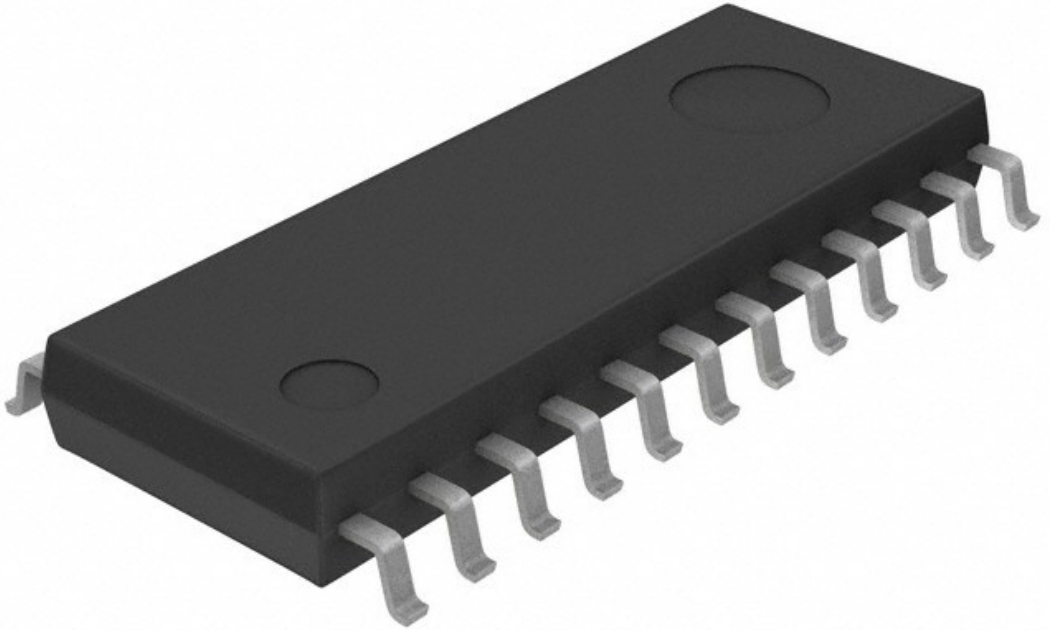
DIP Socket



IC Packaging

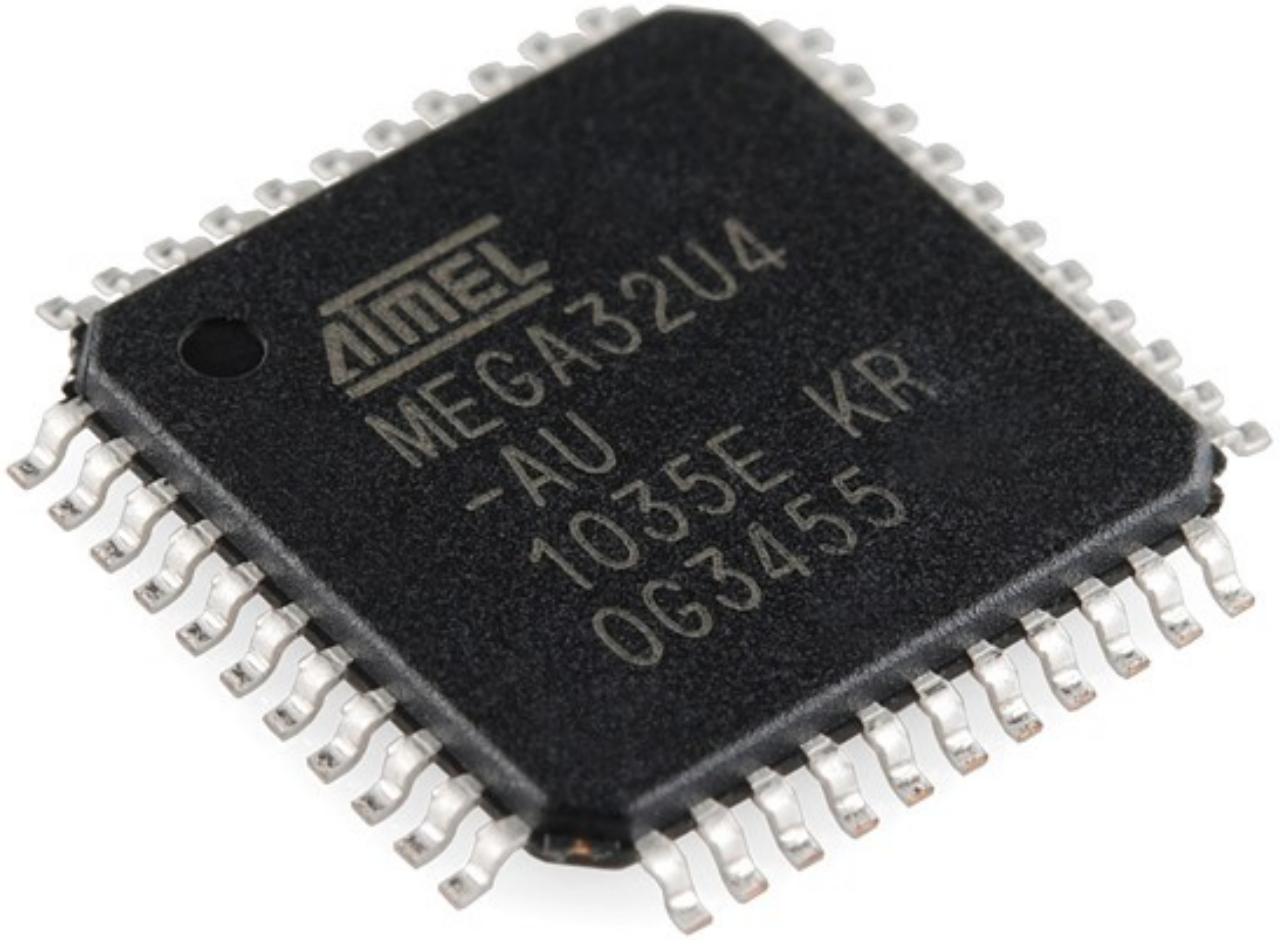
SOP / SSOP / TSOP / TSSOP

[Thin] [Shrink] Small Outline Package



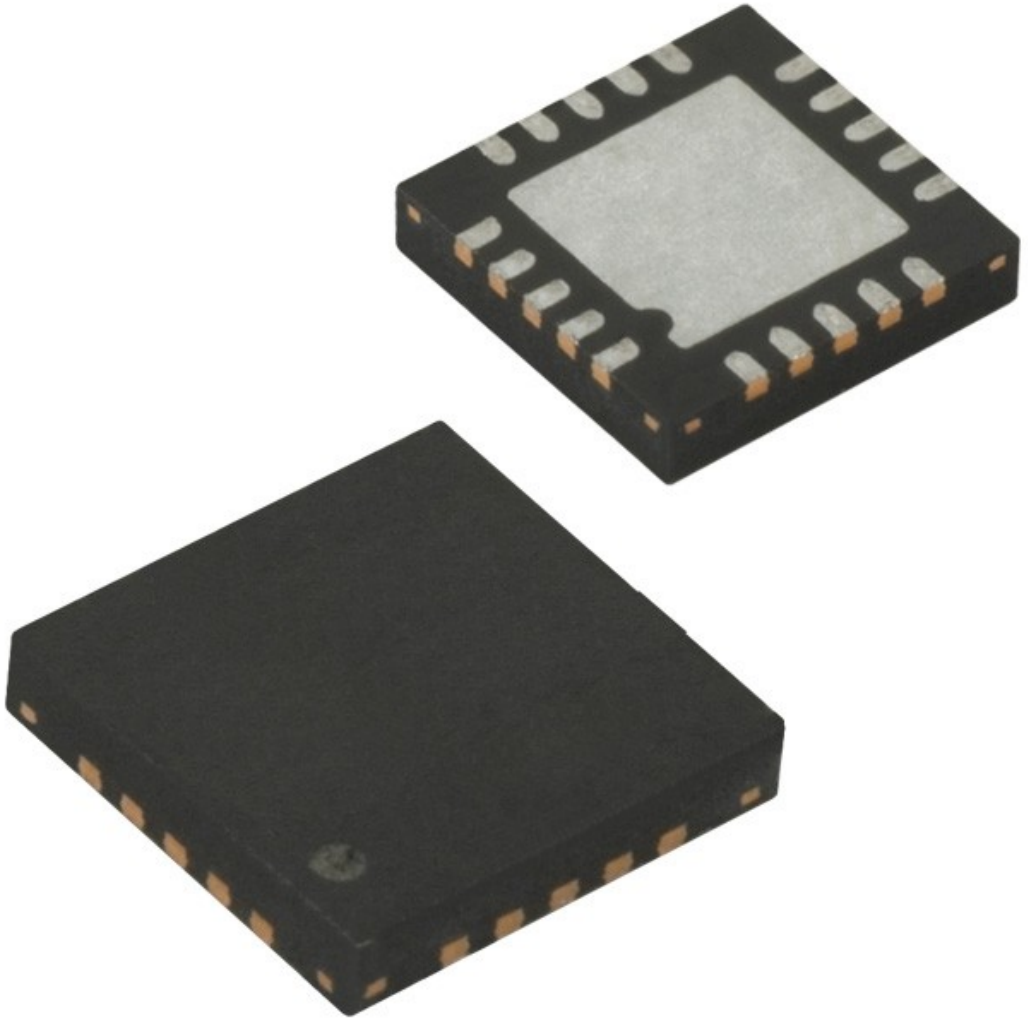
IC Packaging

QFP (Quad Flat Package)



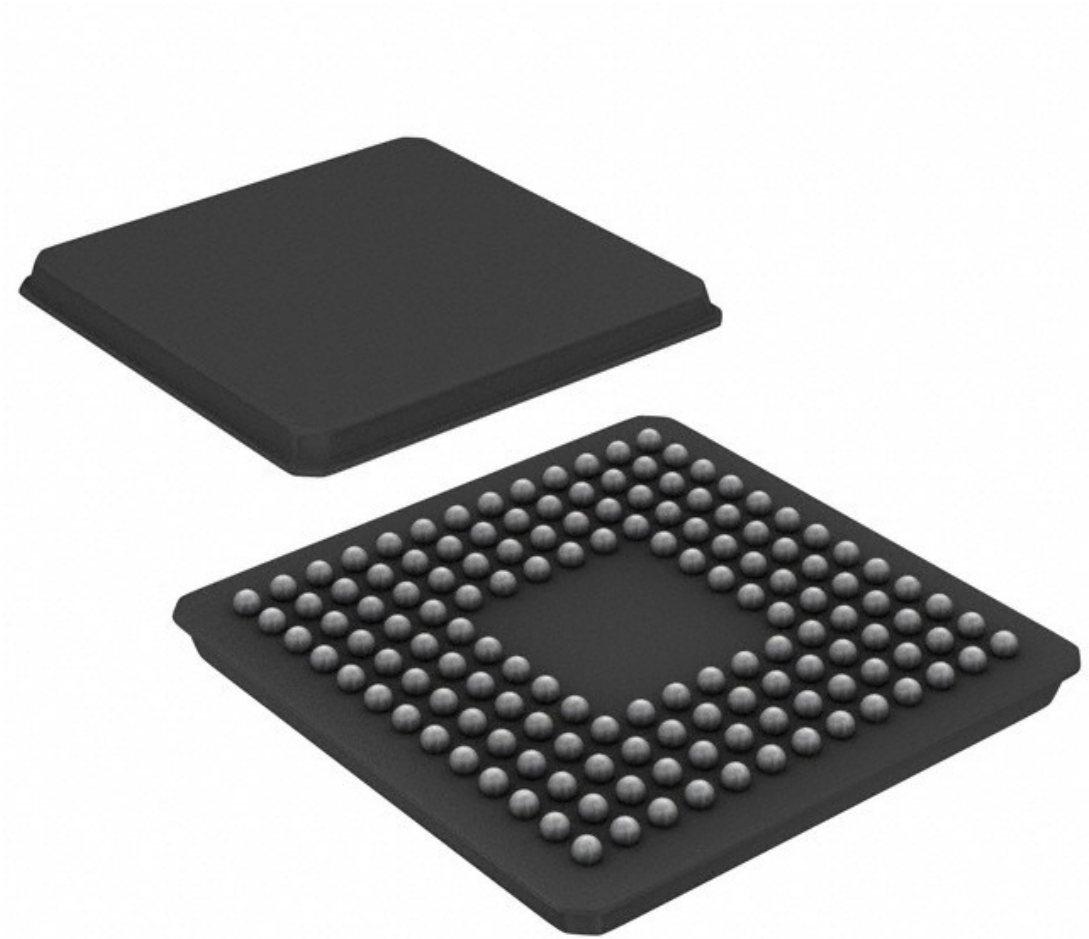
IC Packaging

QFN (Quad Flat No Leads)

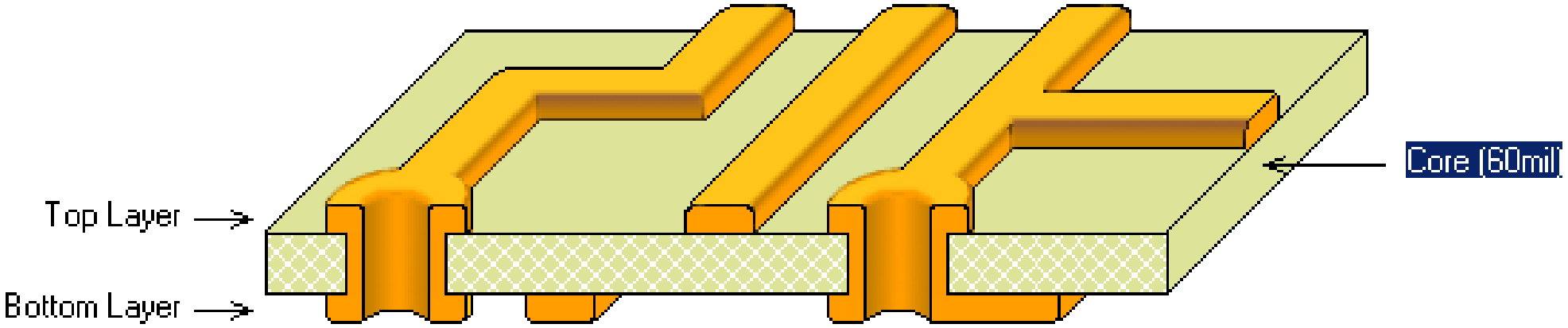


IC Packaging

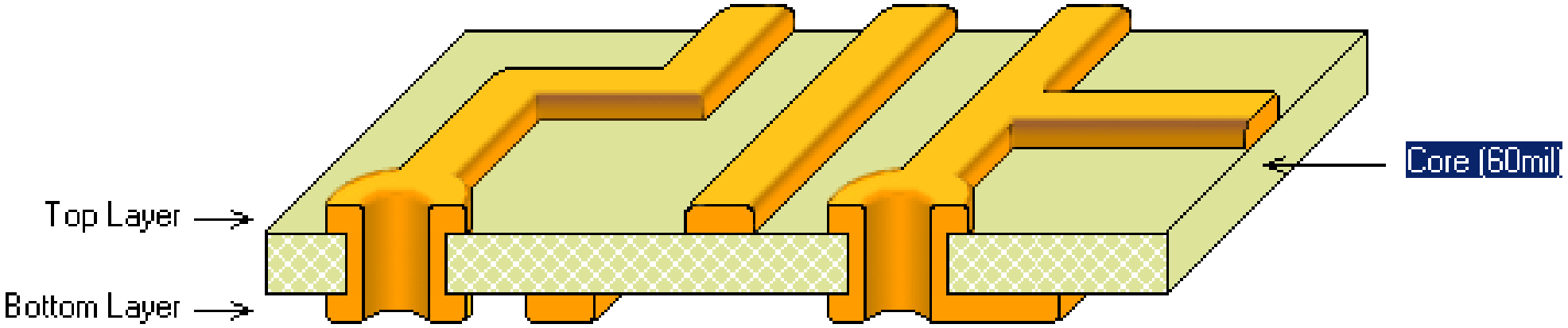
BGA (Ball Grid Array)



Makings of a PCB



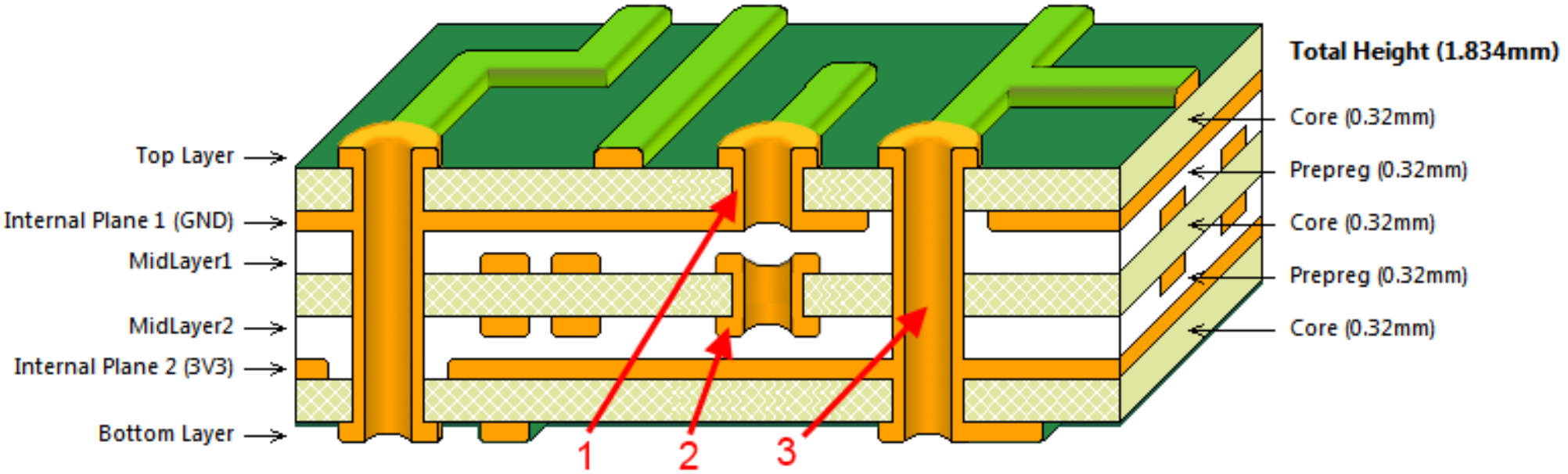
Makings of a PCB



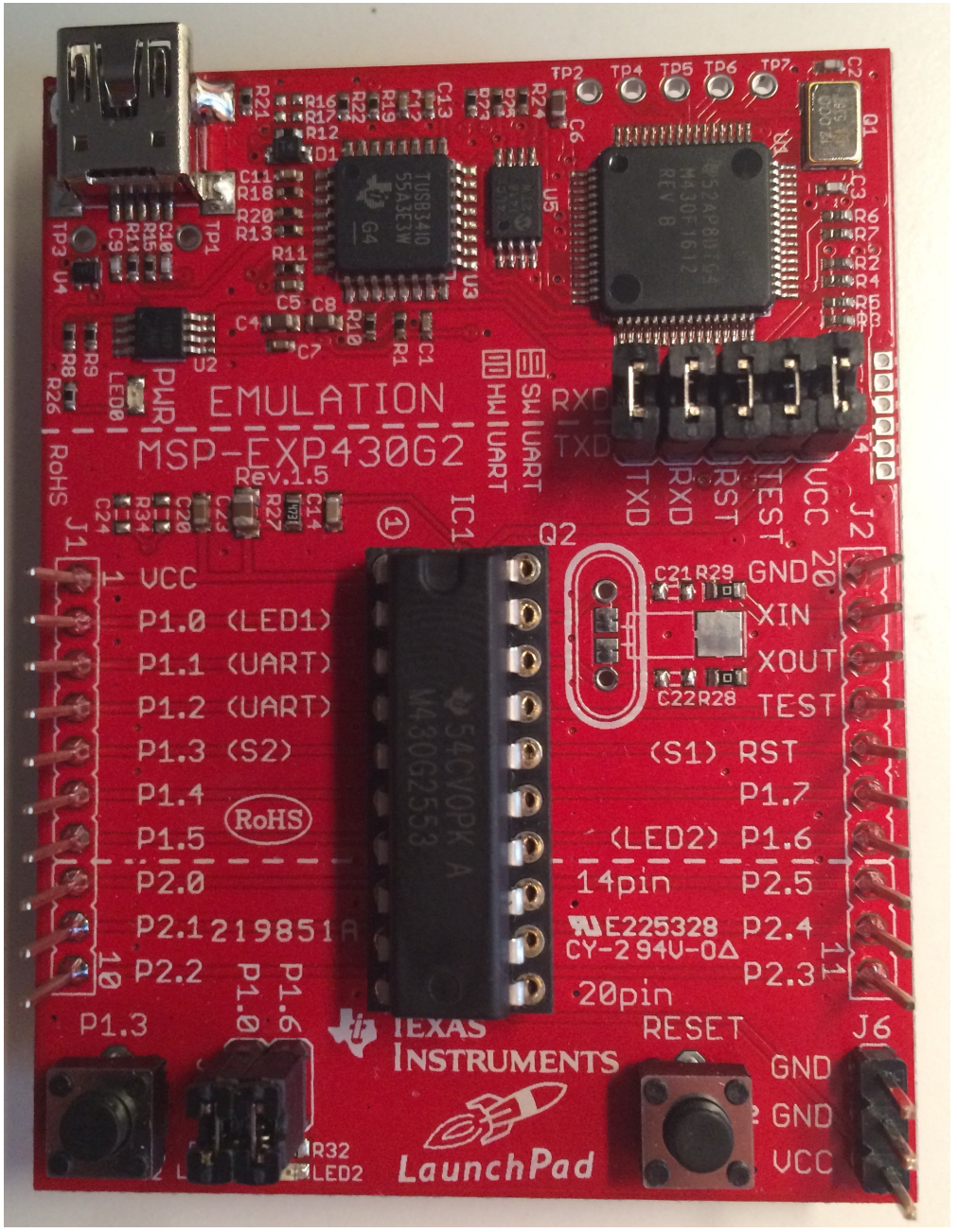
Plus

- Soldermask
- Silk Screen

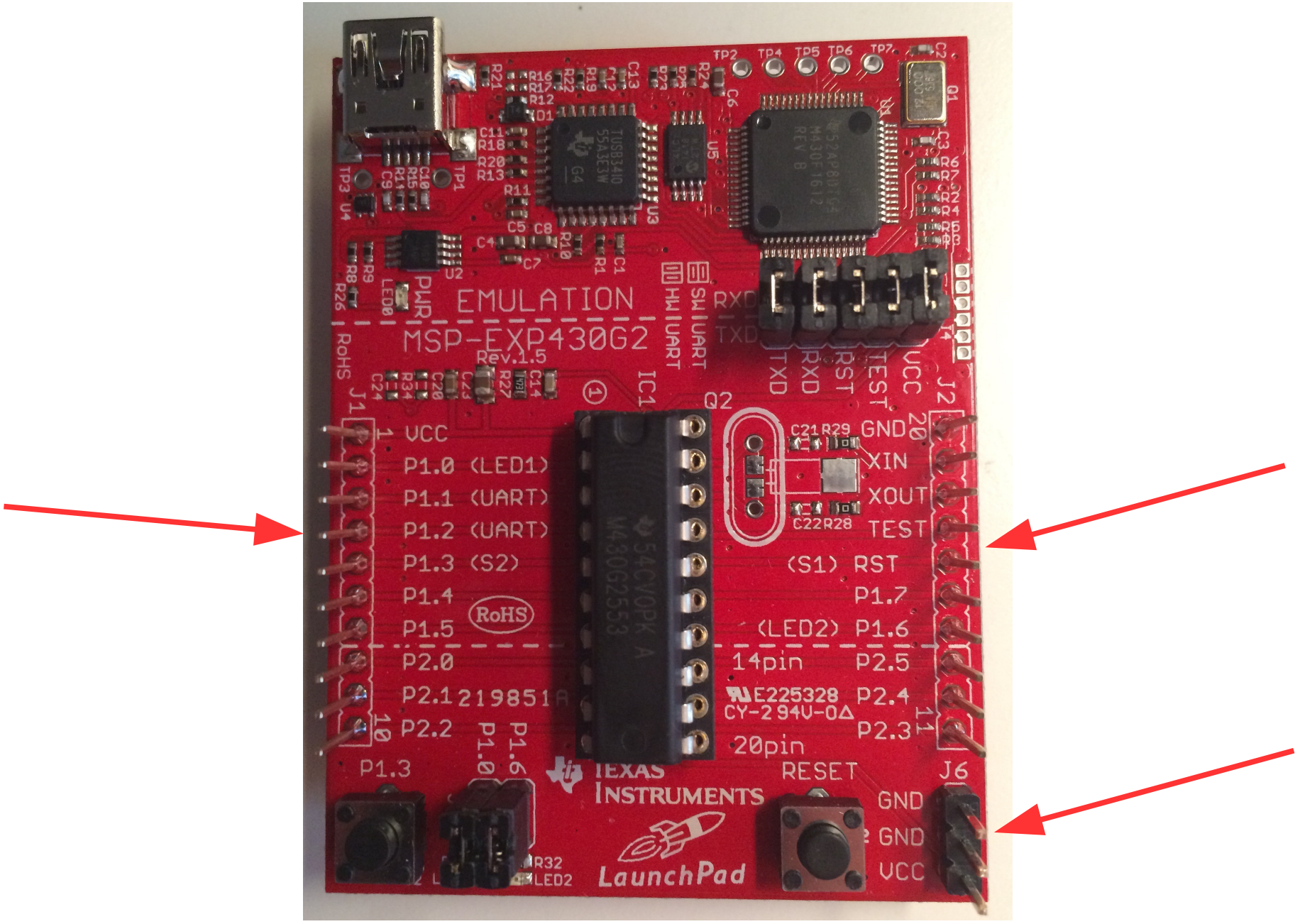
Makings of a PCB



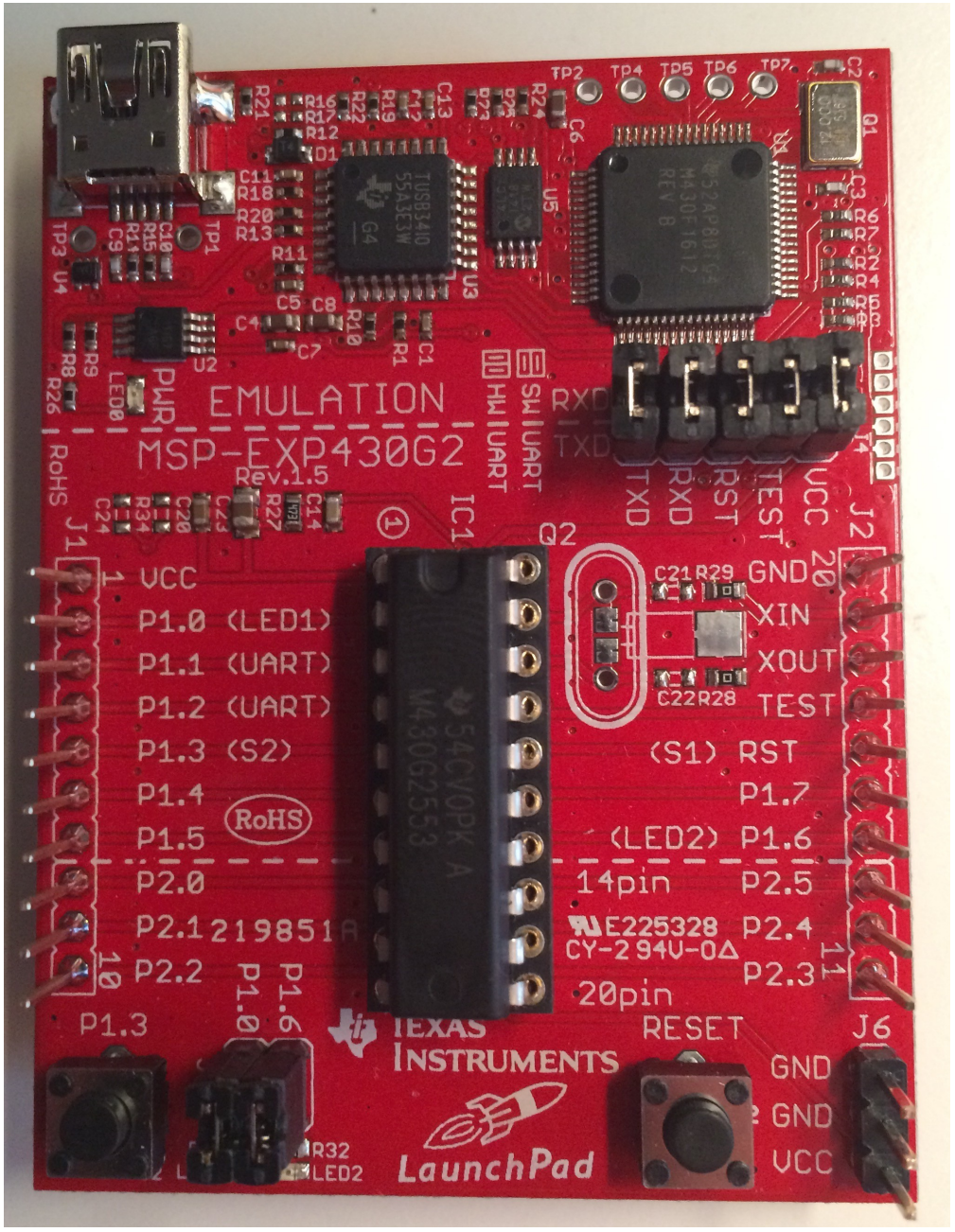
Headers (Populated)



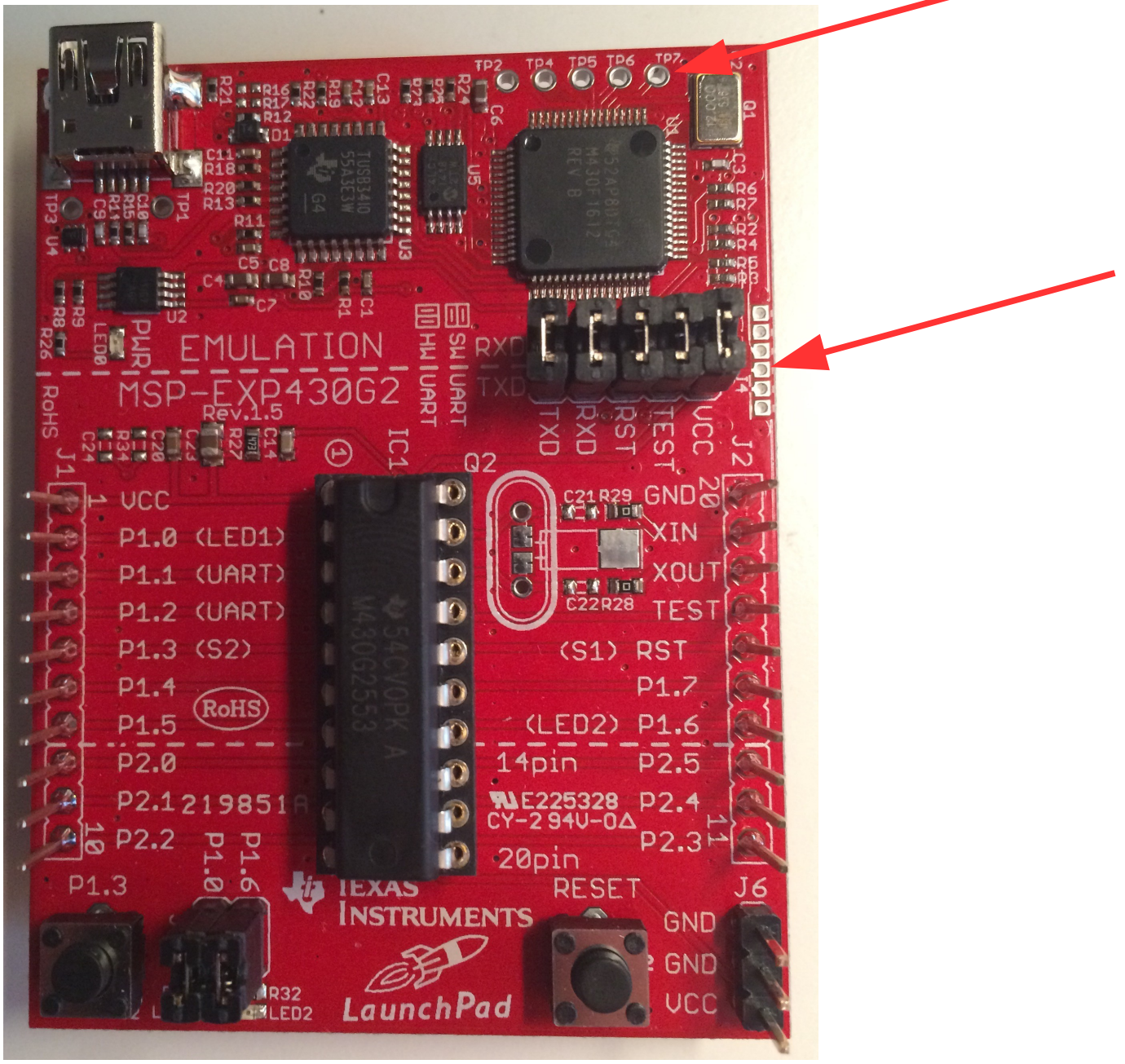
Headers (Populated)



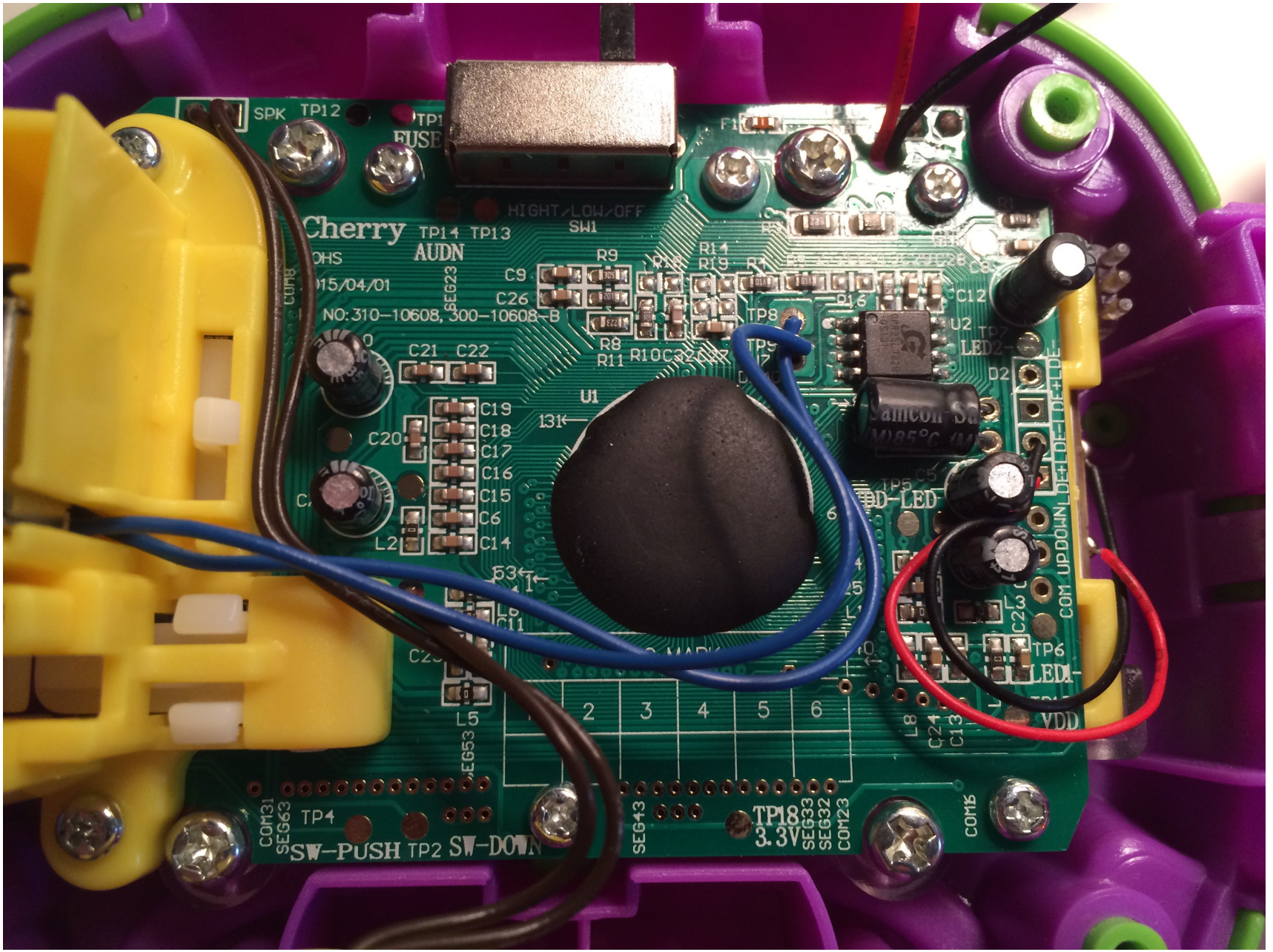
Headers (Unpopulated)



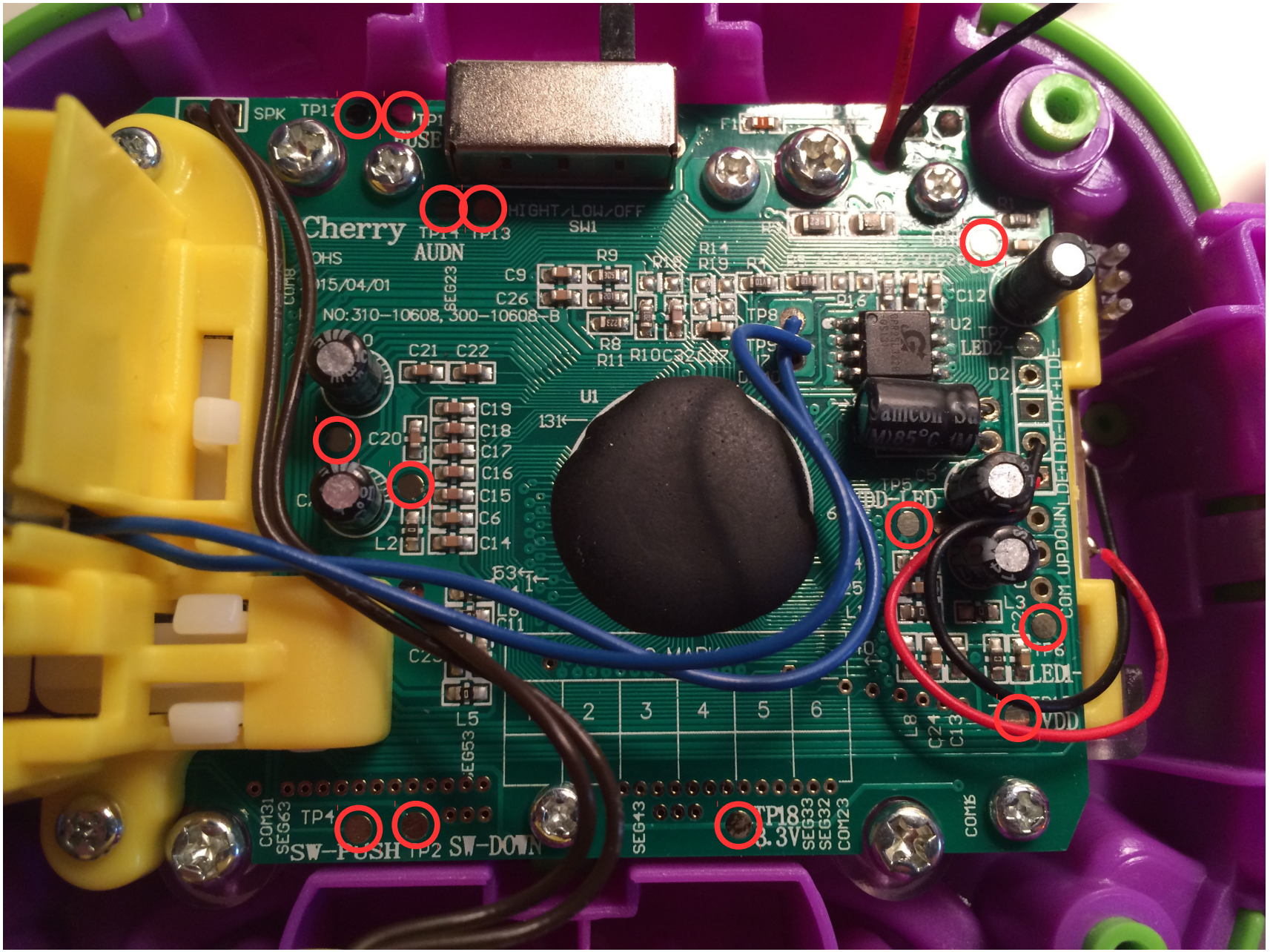
Headers (Unpopulated)



Test Points


SPK
TP12
FUSE
HIGHT/LOW/OFF
SW1
Cherry
TP14
TP13
AUDN
SEG23
C9
C26
R9
R14
R19
C12
TP8
U2
LED2
D2
C21
C22
R8
R11
U1
C19
C18
C17
C16
C15
C6
C14
C20
L2
TP5
LED
UPDOWN
TP6
LED1
VDD
TP4
SW-PUSH
TP2
SW-DOWN
TP18
3.3V
SEG33
SEG32
COM23
SEG43
COM16
COM31
SEG63
SEG53

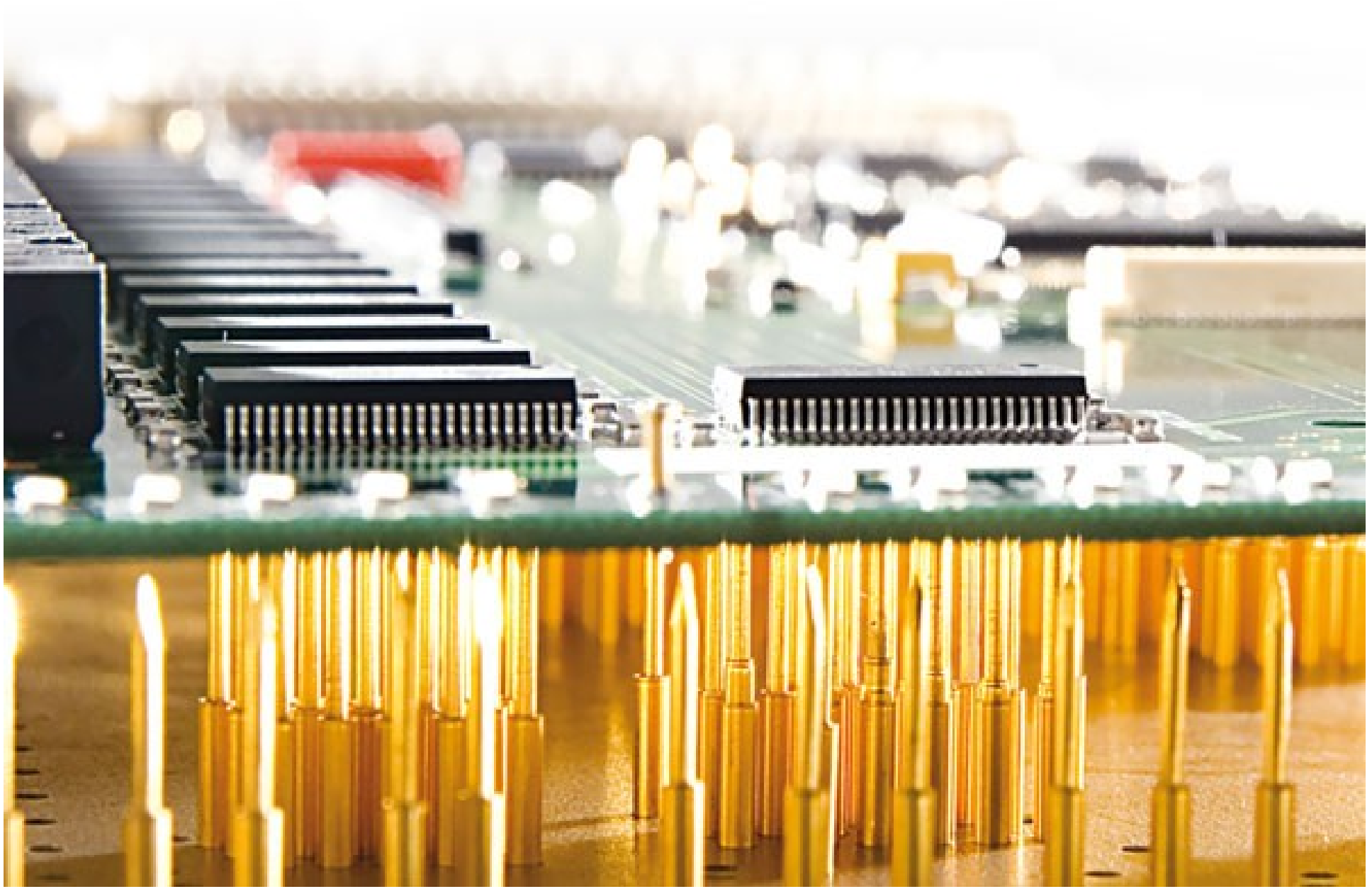
Test Points


SPK
Cherry
AUDN
SW1
U1
U2
LED2-
LED1-
VDD
SW-PUSH
SW-DOWN
TP18
3.3V

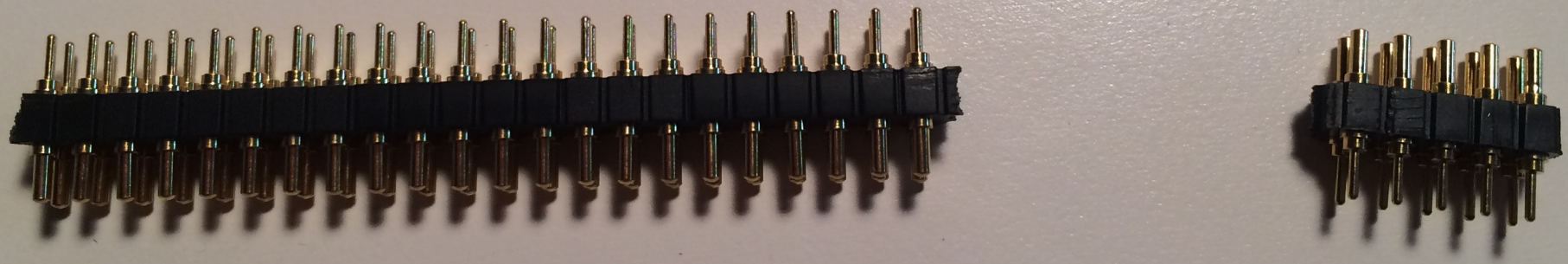
Test Points

- In Circuit Tests (ICT)
- Programming

Bed of Nails



Pogo Pins

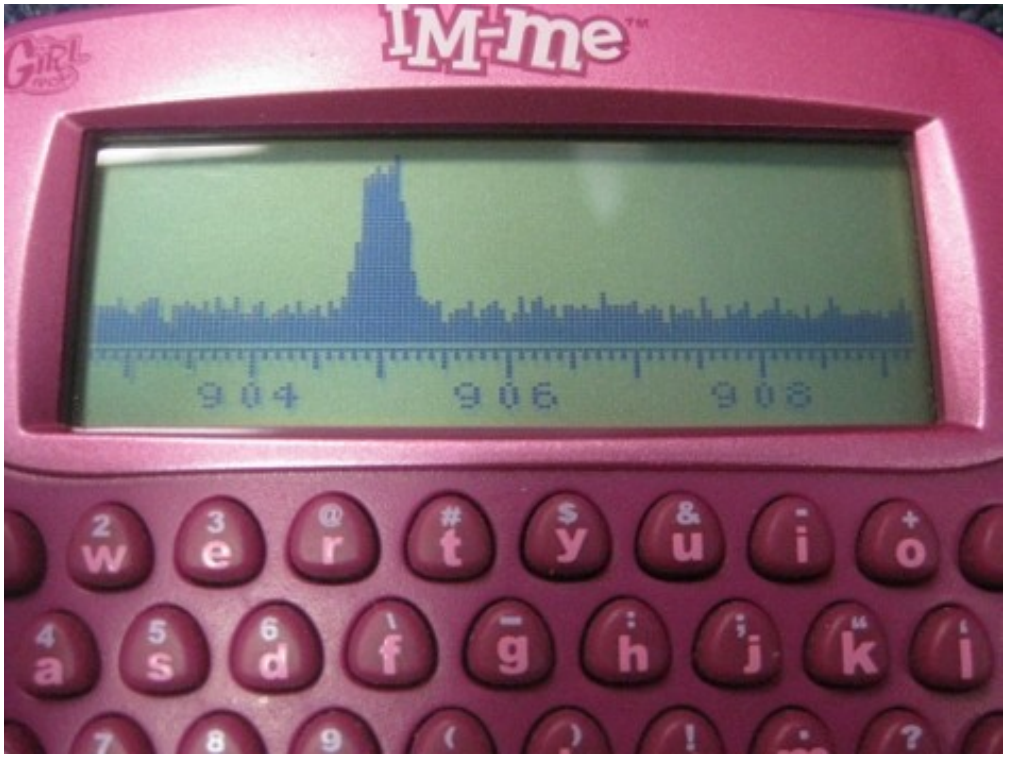


IM Me


IM-me
GirlTech: This SAT
Me: How many friends
did U invite ?
Me: What do U realy want?
GirlTech: Not sure
GirlTech
Space
BACK

IM Me


IM-me
GirlTech: This SAT
Me: How many friends
did U invite?
Me: What do U realy want?
GirlTech: Not sure
GirlTech
Space
BACK


IM-me
904
906
908


FCC ID: PIYL7281H1
QC PASSED C4-002
ULTRA Alkaline
ToysRUs
ULTRA Alkaline
ToysRUs
15AU LR6 1.5V
1.5V
LR6 / AA

GFC-14P

GIMME

YES

NO

Supply power from GoodFET to target?

5: GND

4: 3V3

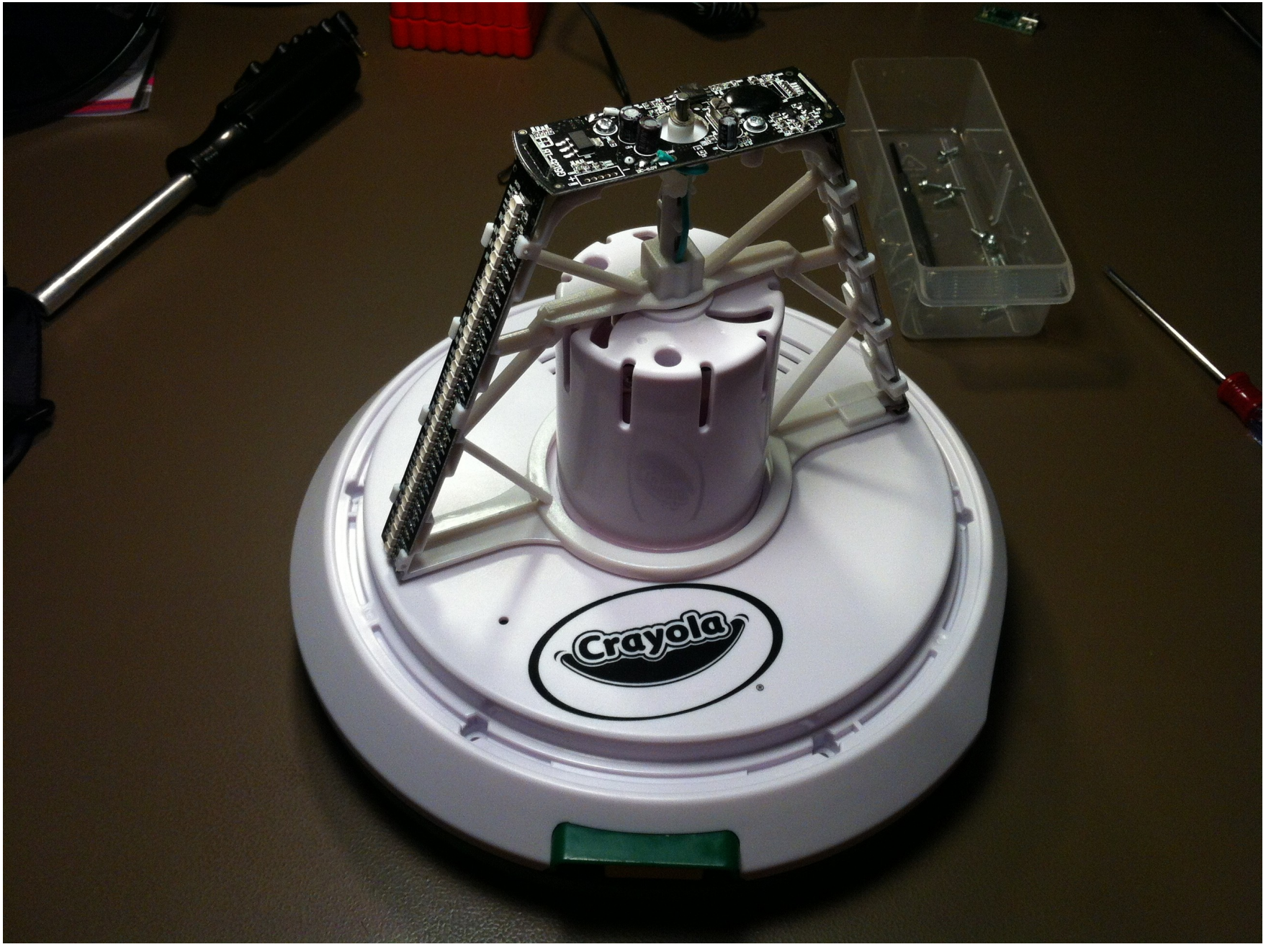
3: DC/P2_2

2: DD/P2_1

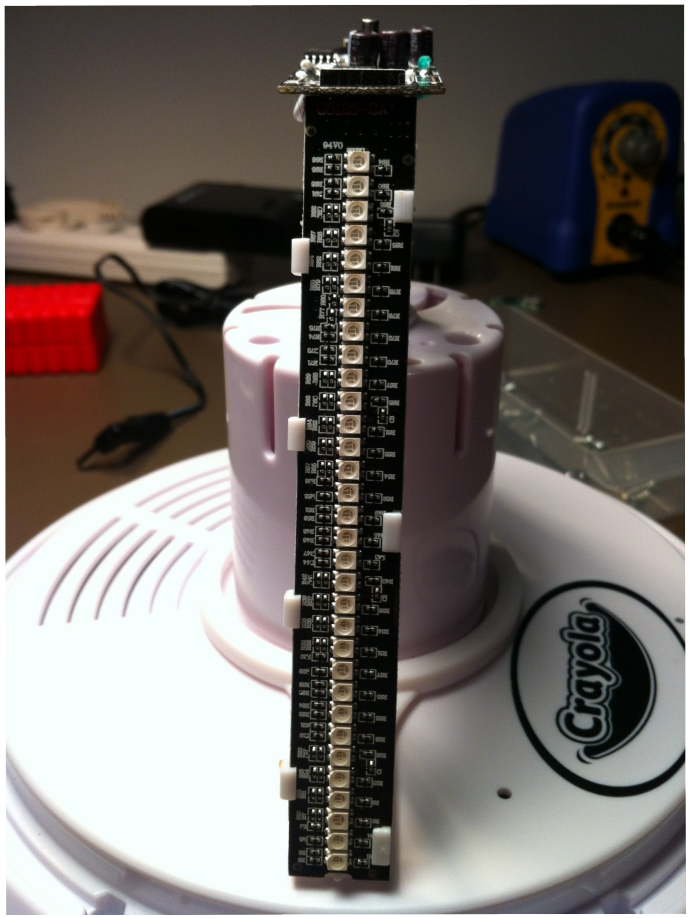
1: RESET_N

Crayola Digital Light Designer

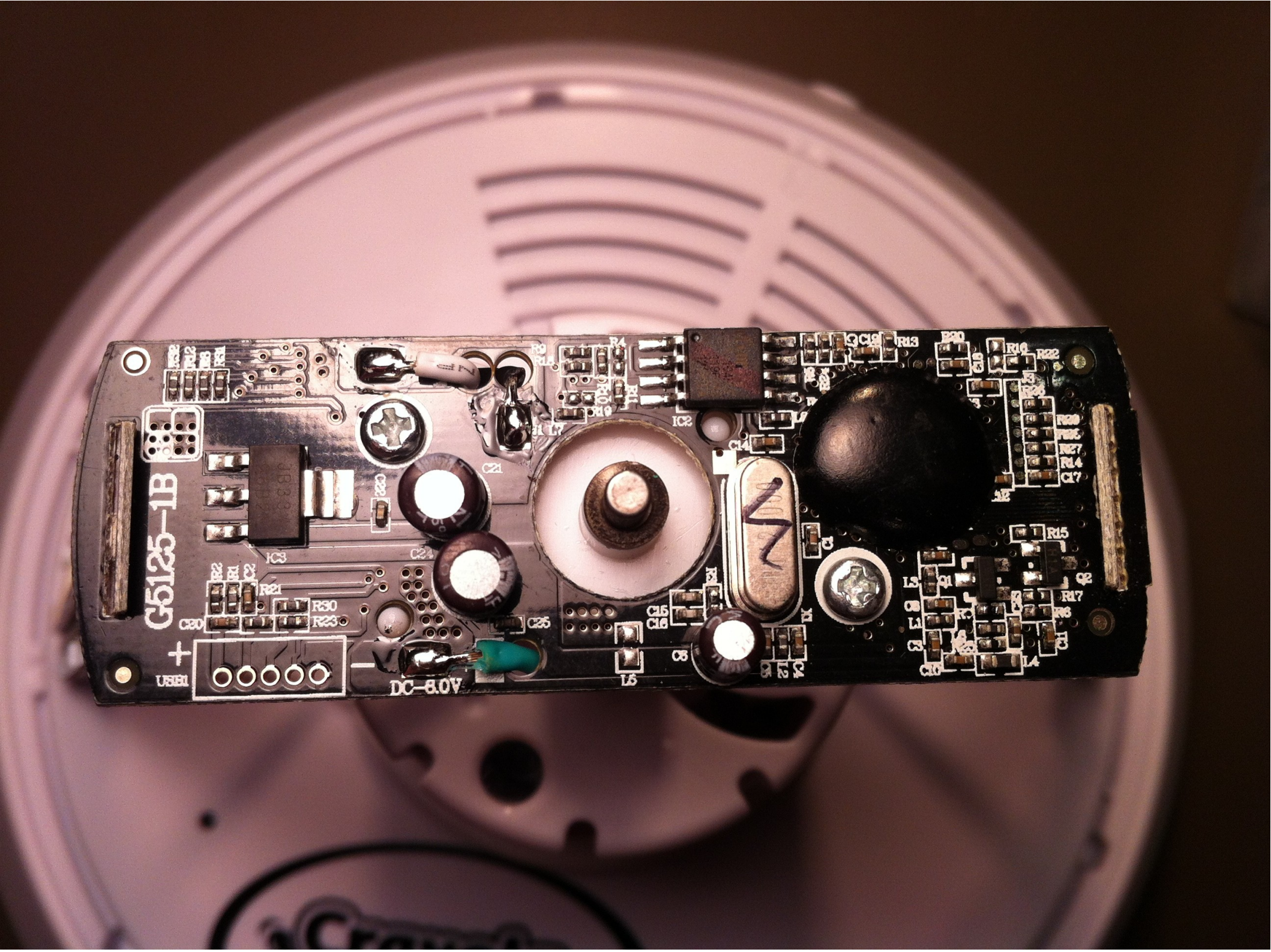
Crayola Digital Light Designer

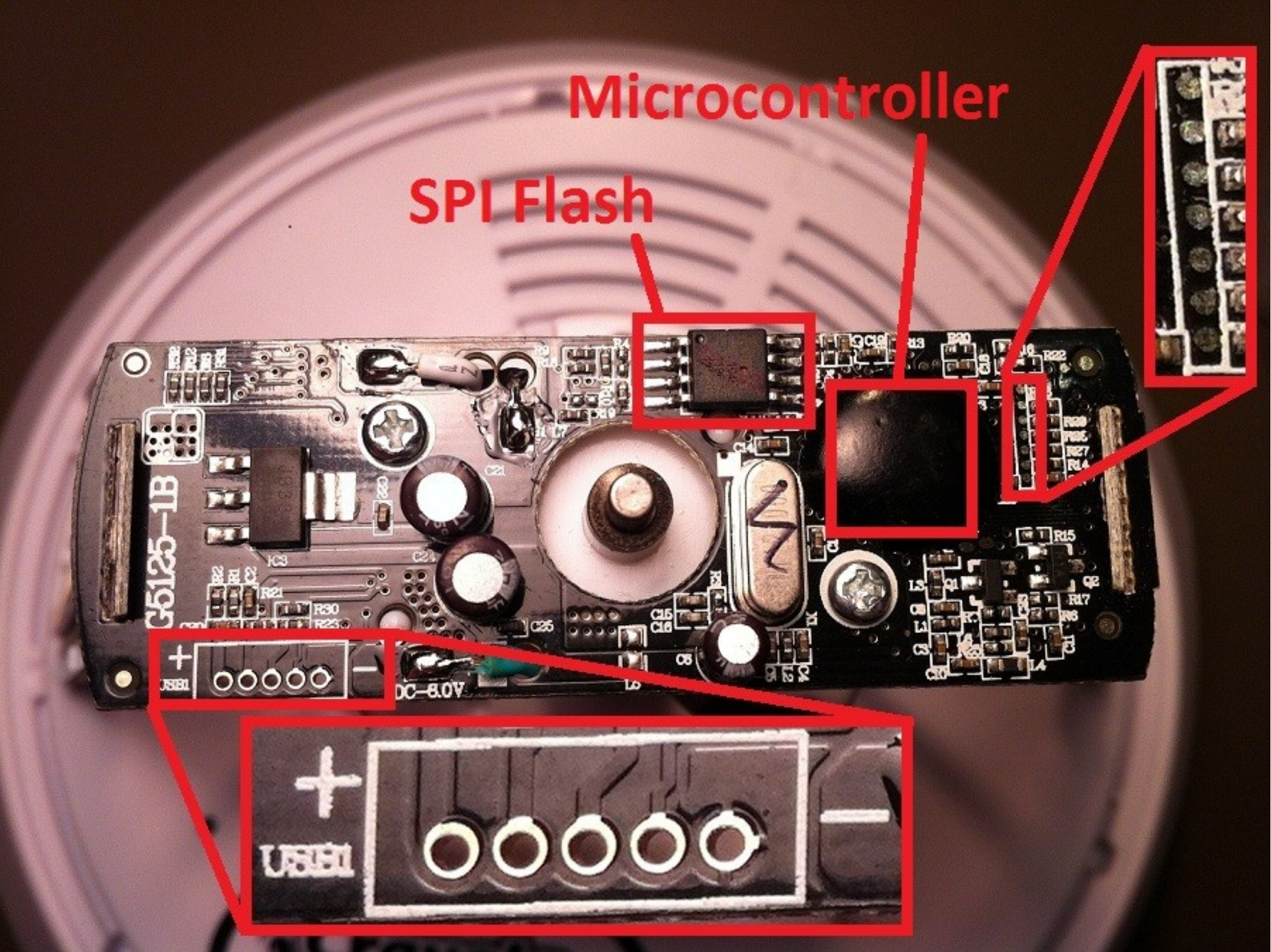


Crayola Digital Light Designer

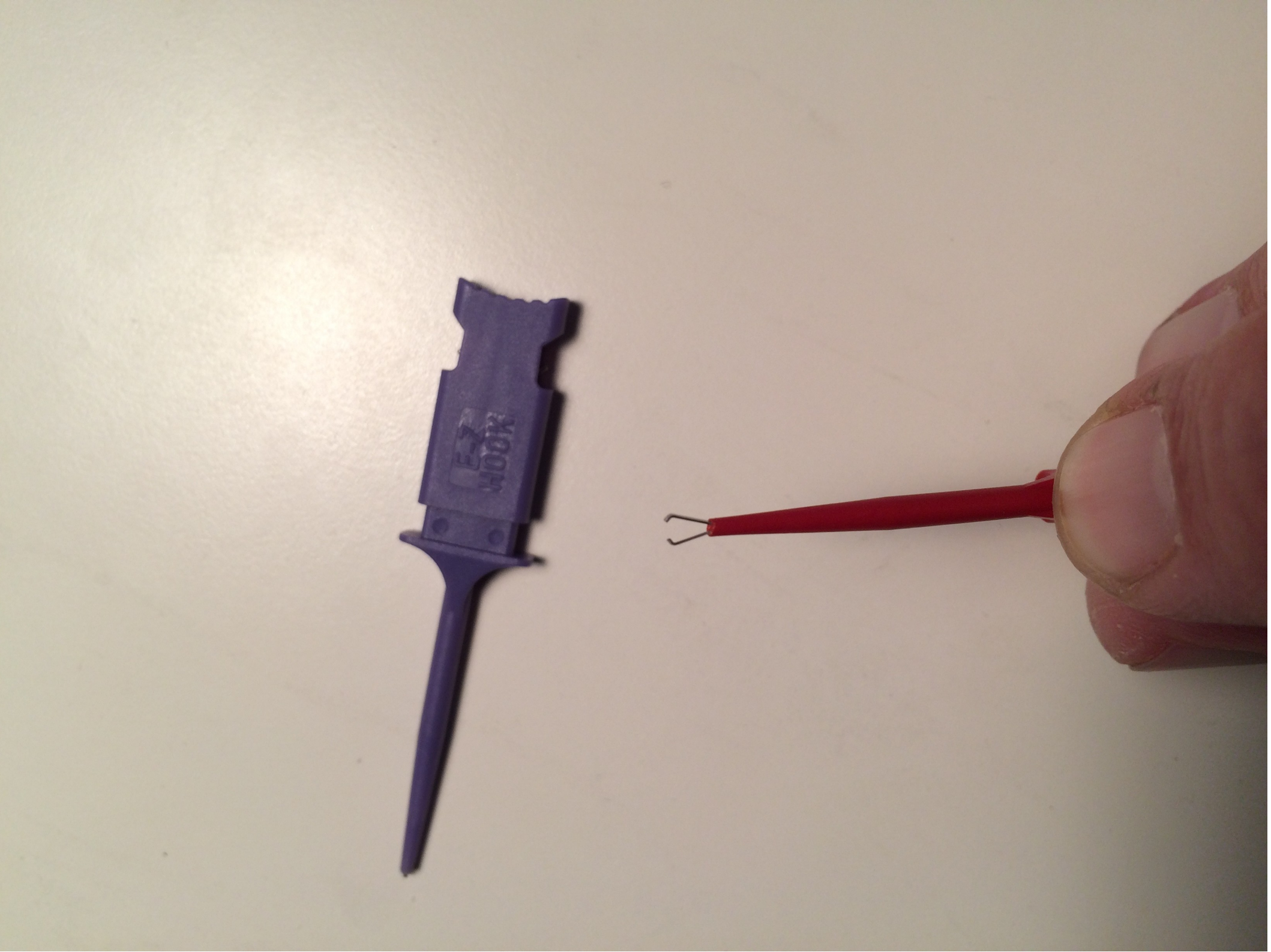


- 32 RGB LEDs
- 16 photo resistors (IR)
- Wand has IR LED at tip

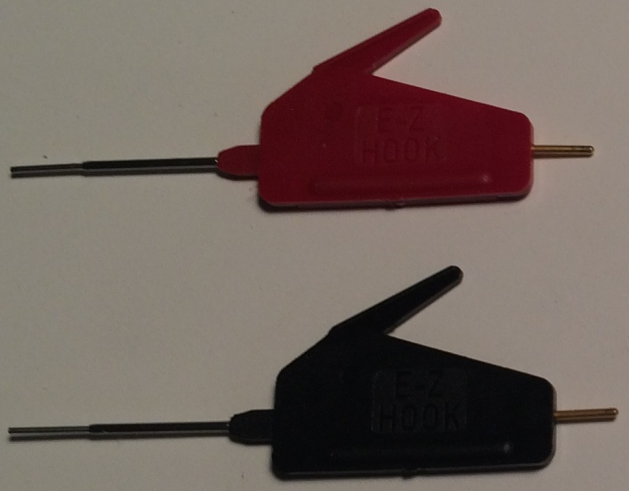

G5125-1B
USB1
DC-6.0V


Microcontroller
SPI Flash
G5125-1B
USB1
DC-5.0V

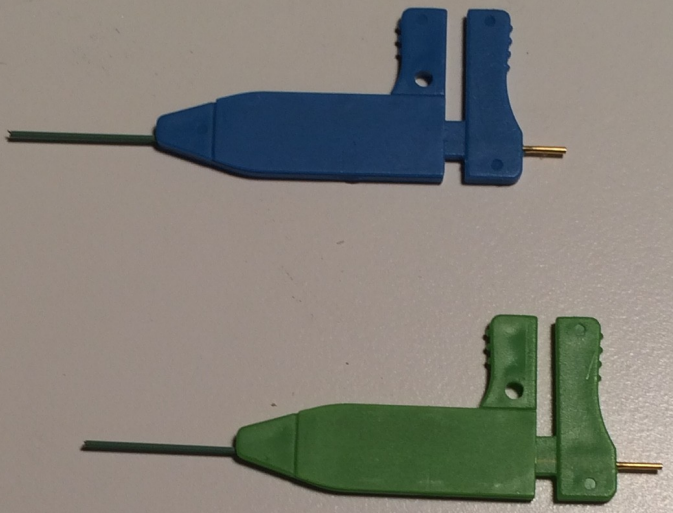
XKM Grabbers


E-Z
HOOK

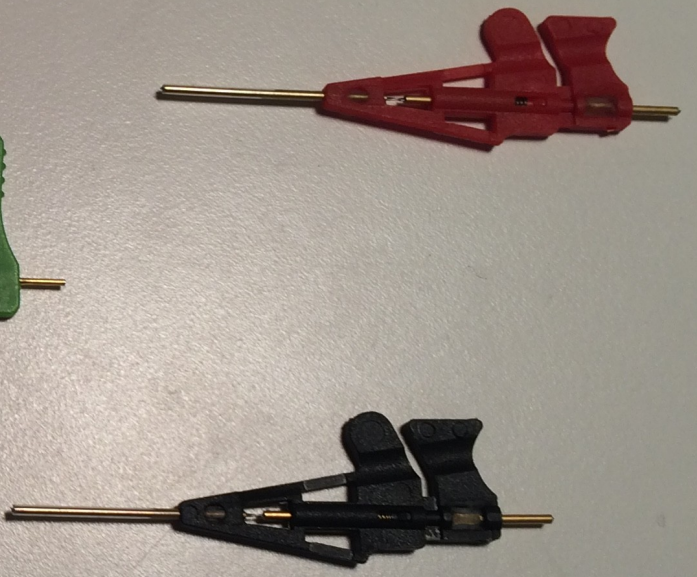
Even Smaller Grabbers



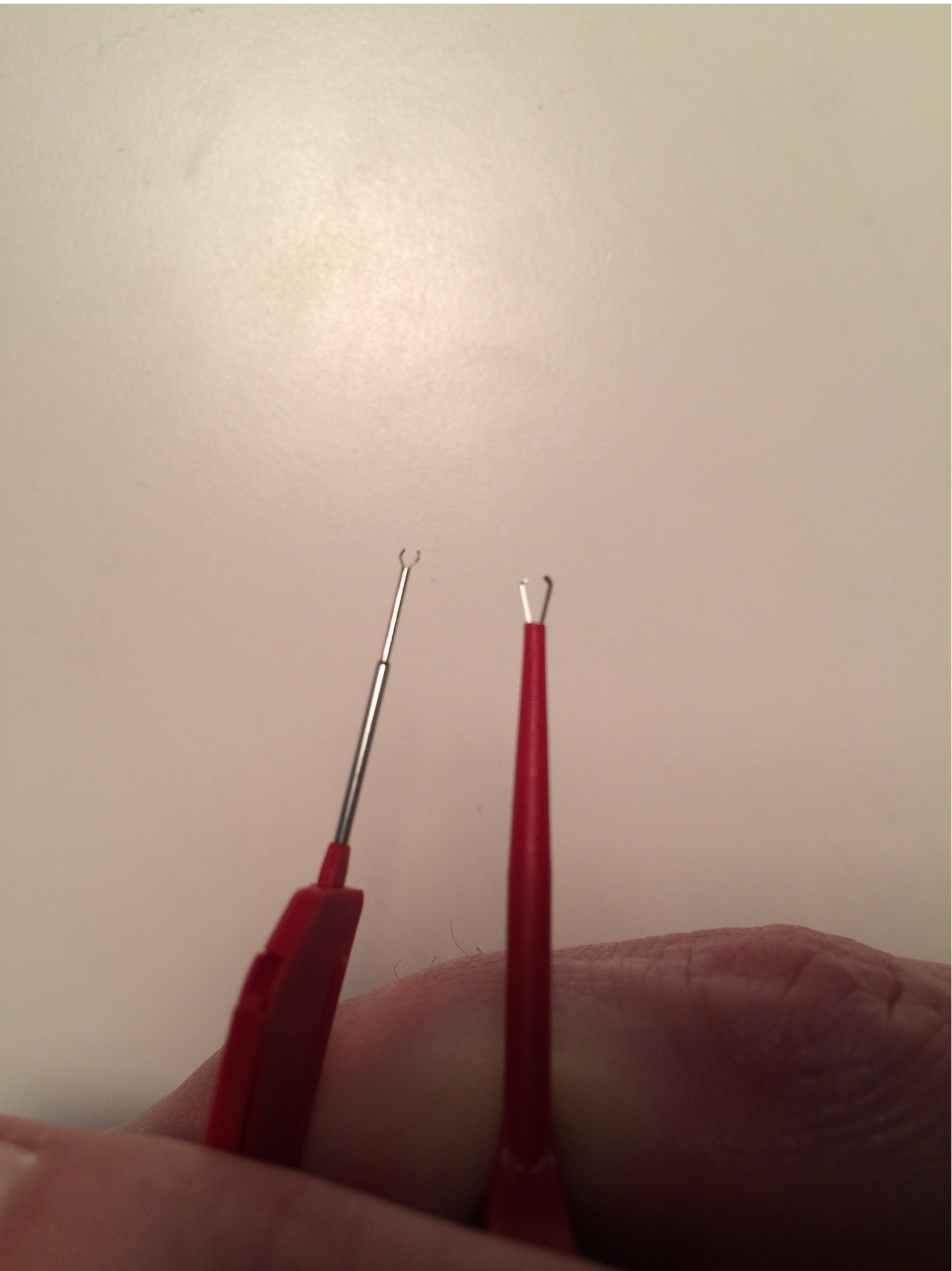
EZ-Hook X2015



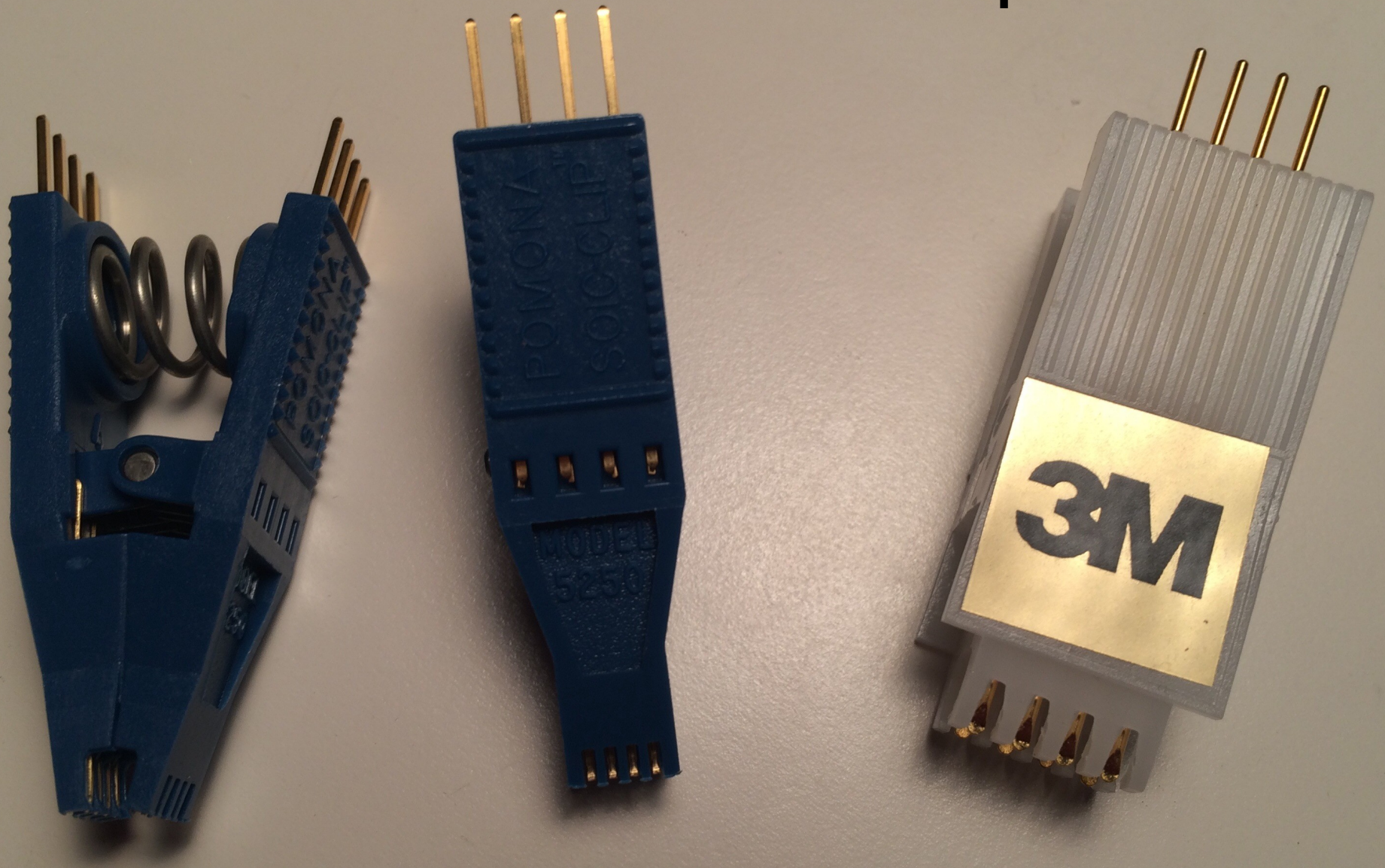
TPI NC1



Pomona 72902



Test Clips



Pomona SOP-8

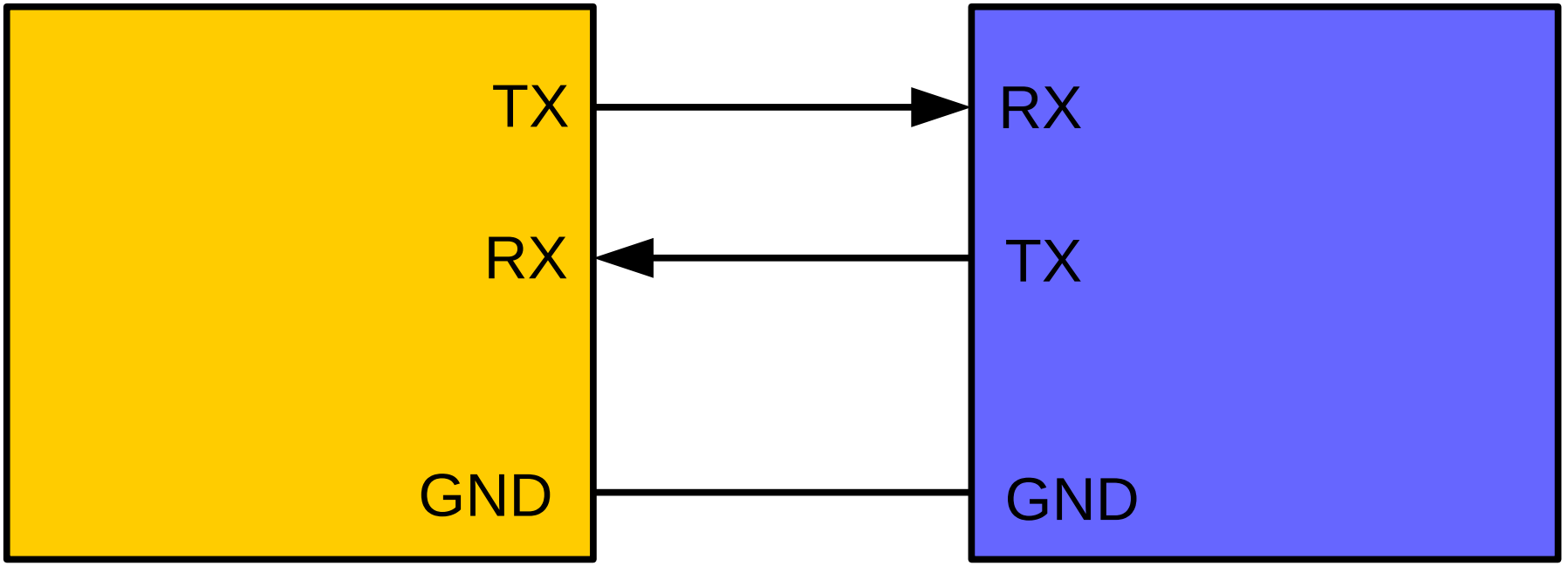
3M DIP-8

Other Methods

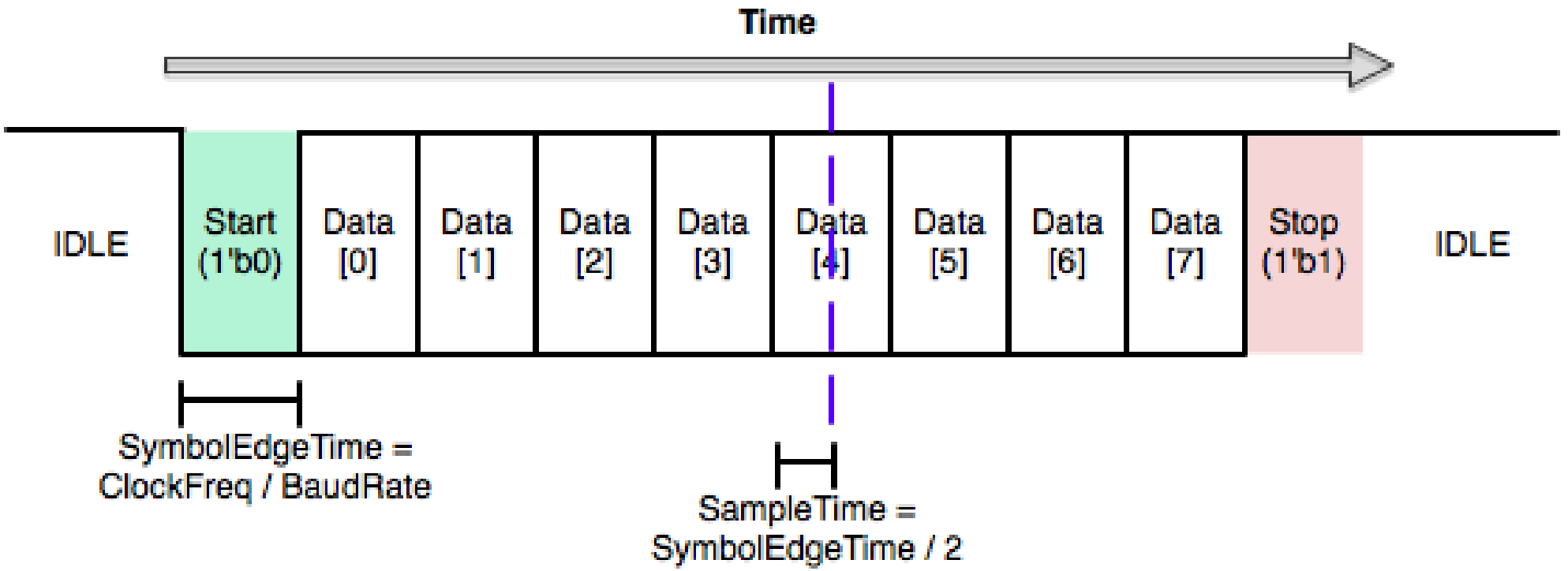
- Solder test wires directly onto board / ICs
- Hypodermic needles

Communication Busses

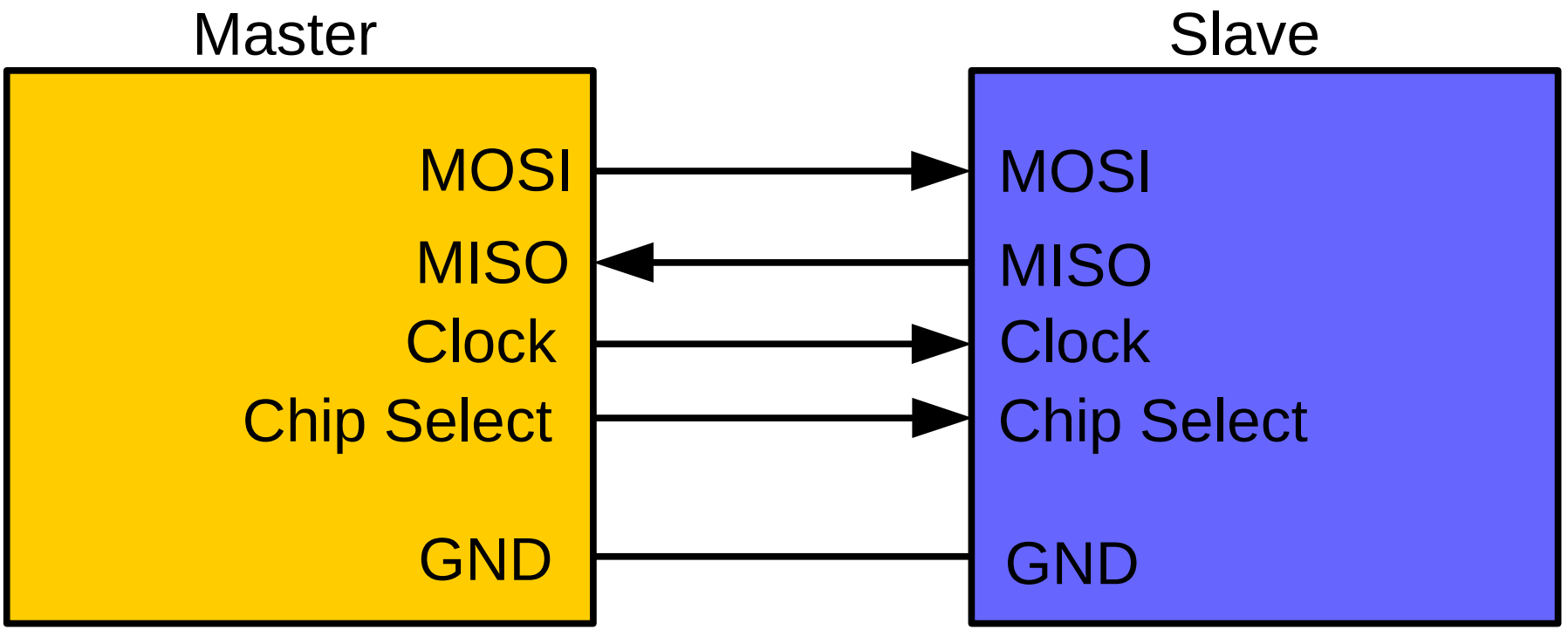
UART

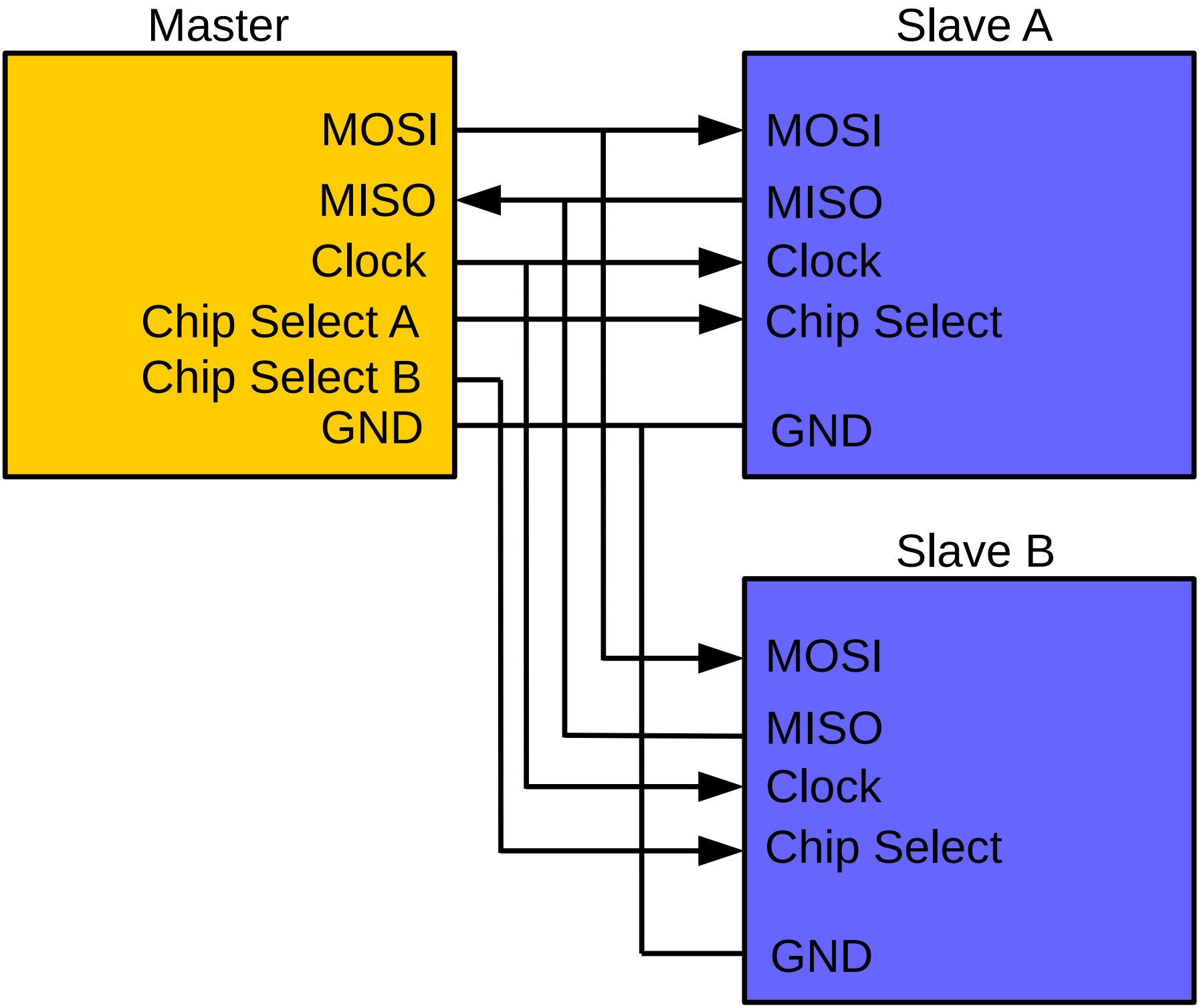


UART

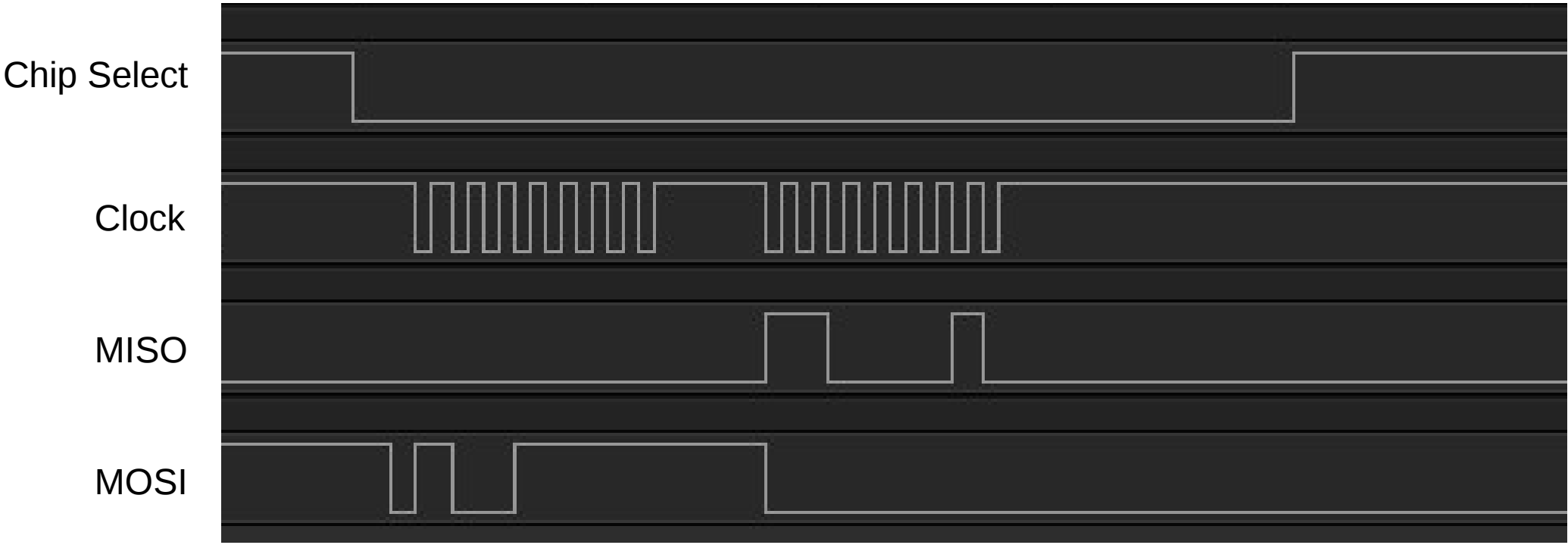

Time
IDLE
Start
(1'b0)
Data
[0]
Data
[1]
Data
[2]
Data
[3]
Data
[4]
Data
[5]
Data
[6]
Data
[7]
Stop
(1'b1)
IDLE
SymbolEdgeTime =
ClockFreq / BaudRate
SampleTime =
SymbolEdgeTime / 2

SPI

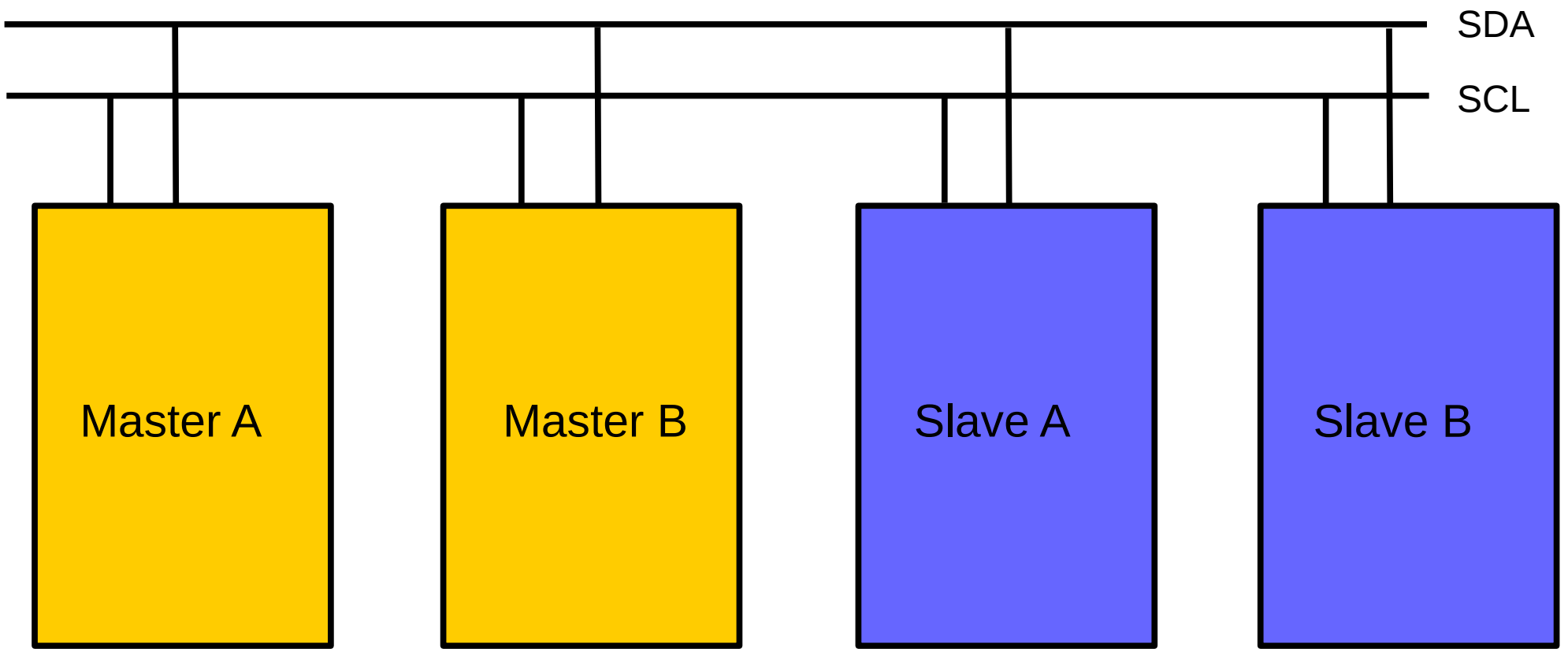

Master
Slave
MOSI
MISO
Clock
Chip Select
GND
MOSI
MISO
Clock
Chip Select
GND


Master
MOSI
MISO
Clock
Chip Select A
Chip Select B
GND
Slave A
MOSI
MISO
Clock
Chip Select
GND
Slave B
MOSI
MISO
Clock
Chip Select
GND

SPI

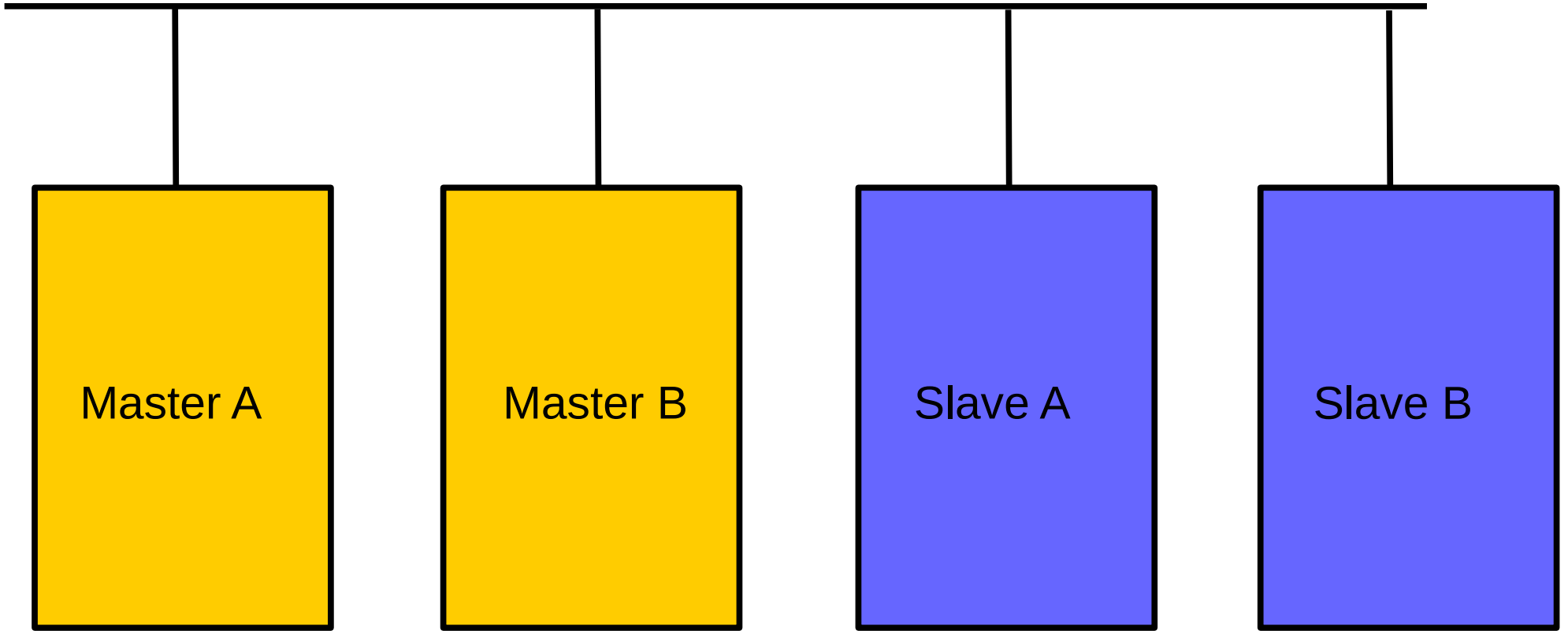


I^2C



SDA : Serial Data
SCL : Serial Clock

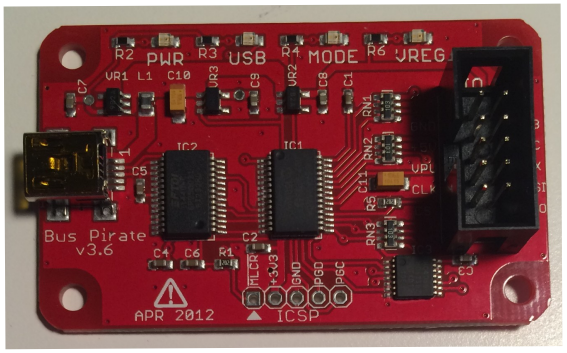
1-Wire



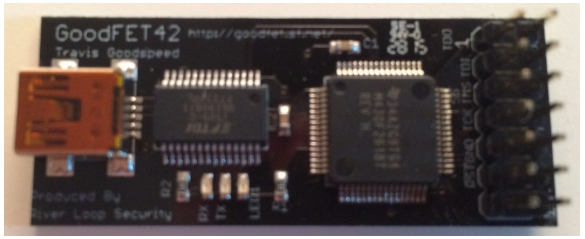
Hardware Hacking Tools

- Provide access to read / write to common communication busses
- JTAG support (more on this later…)
- Easy to use software tools

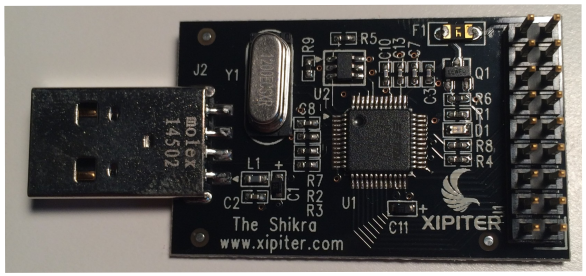
Hardware Hacking Tools



BusPirate
Pros: Cheap
Cons: Buggy, **very** slow

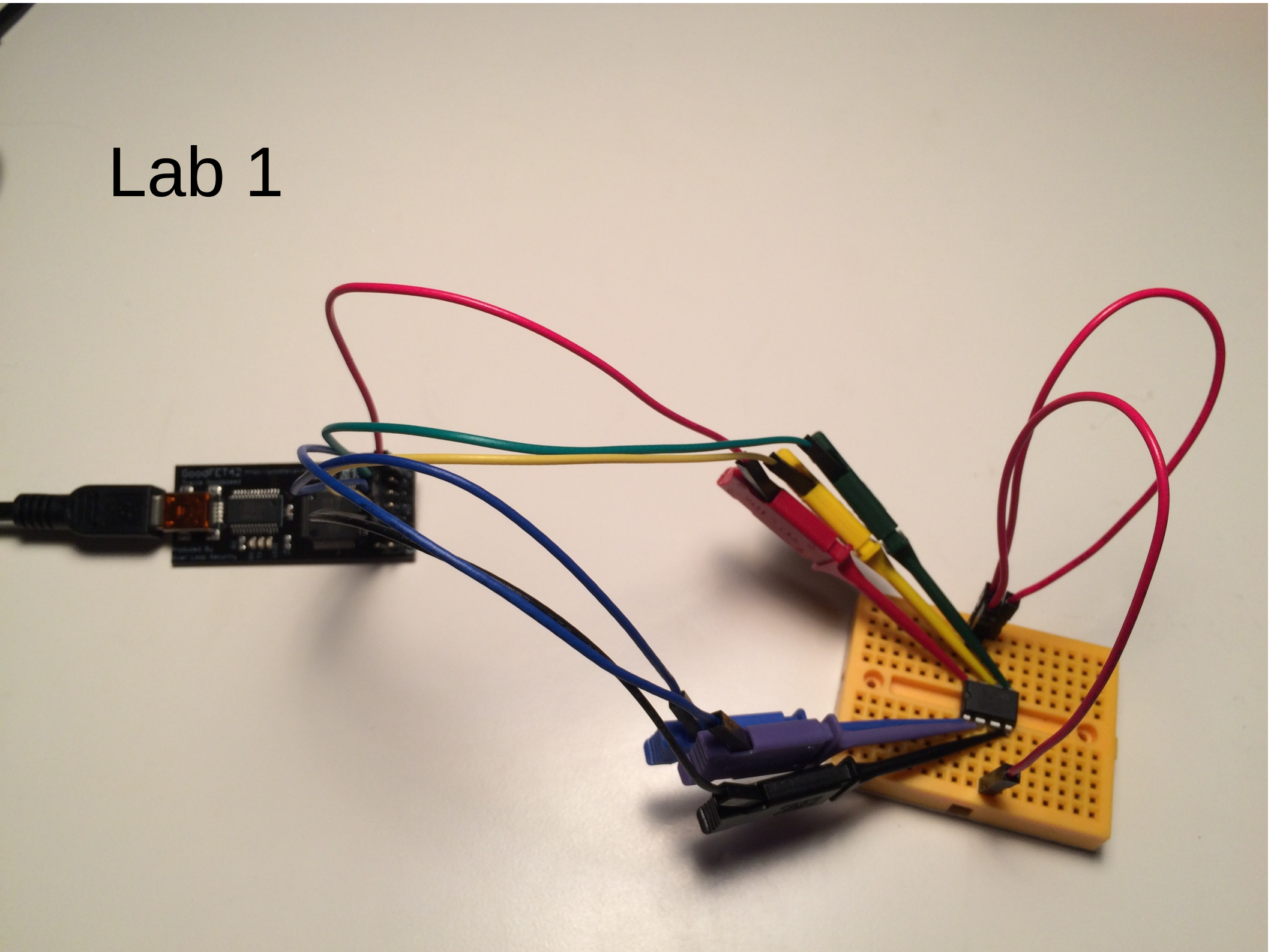


GoodFET
Pros: Fast, versatile
Cons: JTAG debugging is DIY



Xipiter Shikra
Pros: Fast, good JTAG support
Cons: Next to no documentation

Lab 1



Lab 1 – SPI Flash Extraction

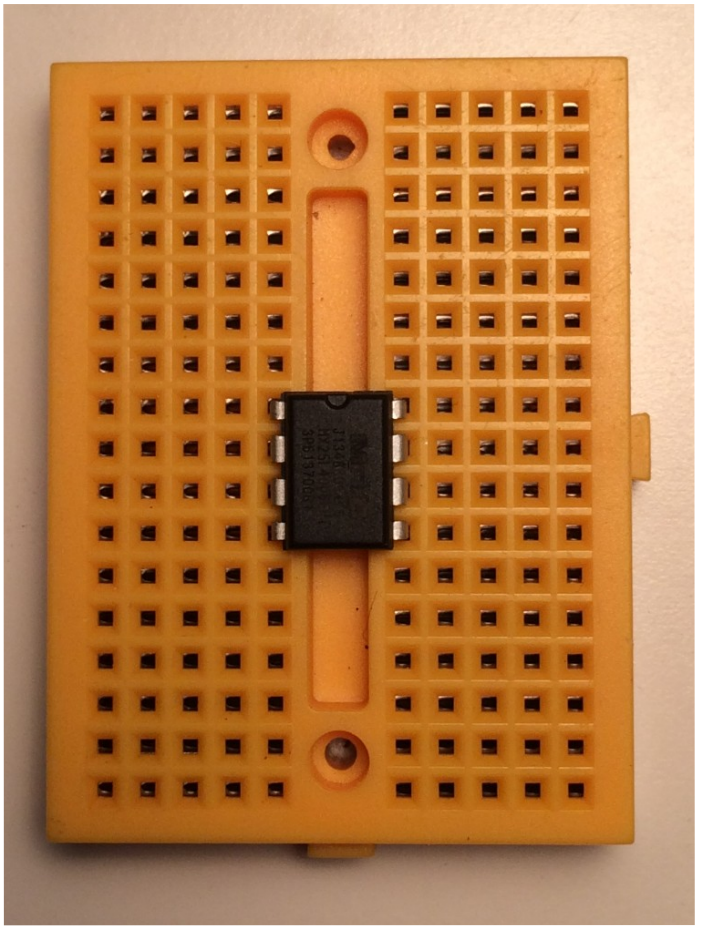
4 Mb Flash IC

Read / Write through SPI

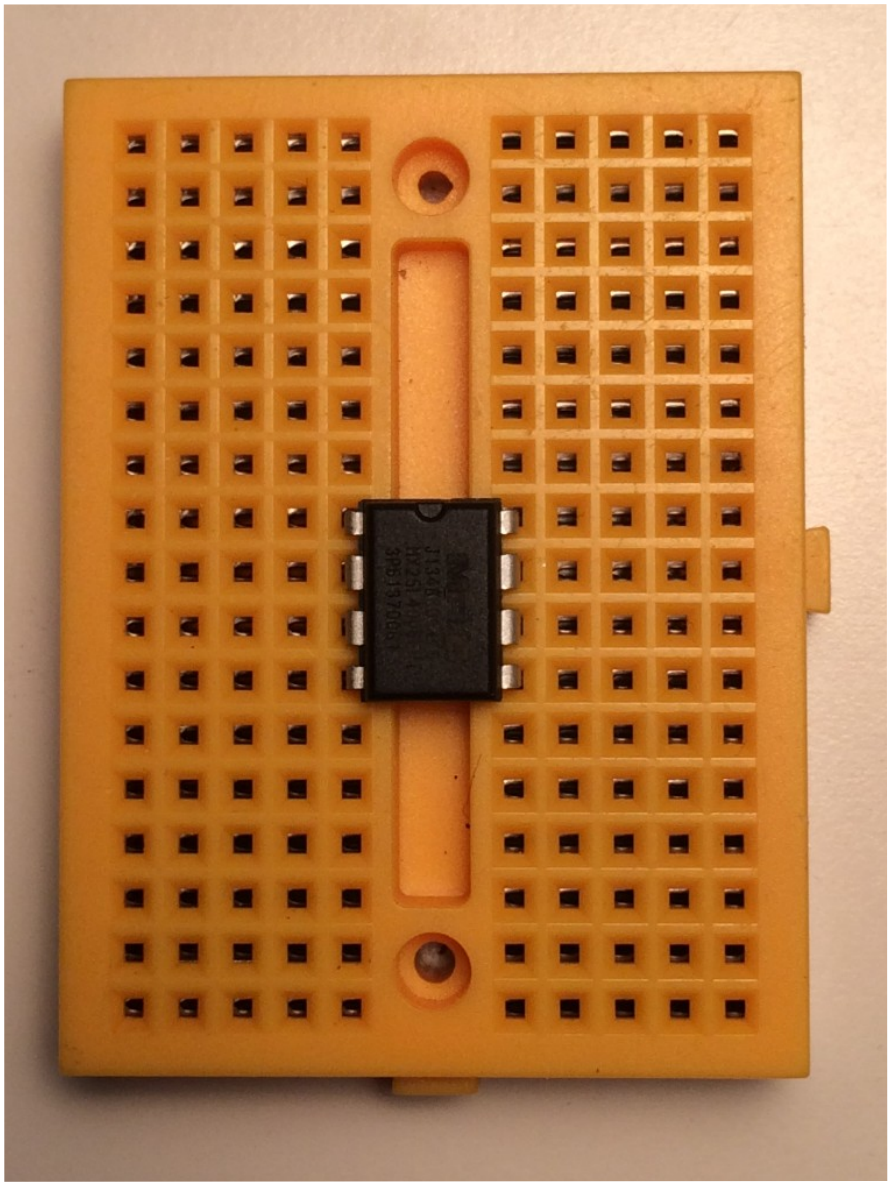
Macronix

MX25L4006E

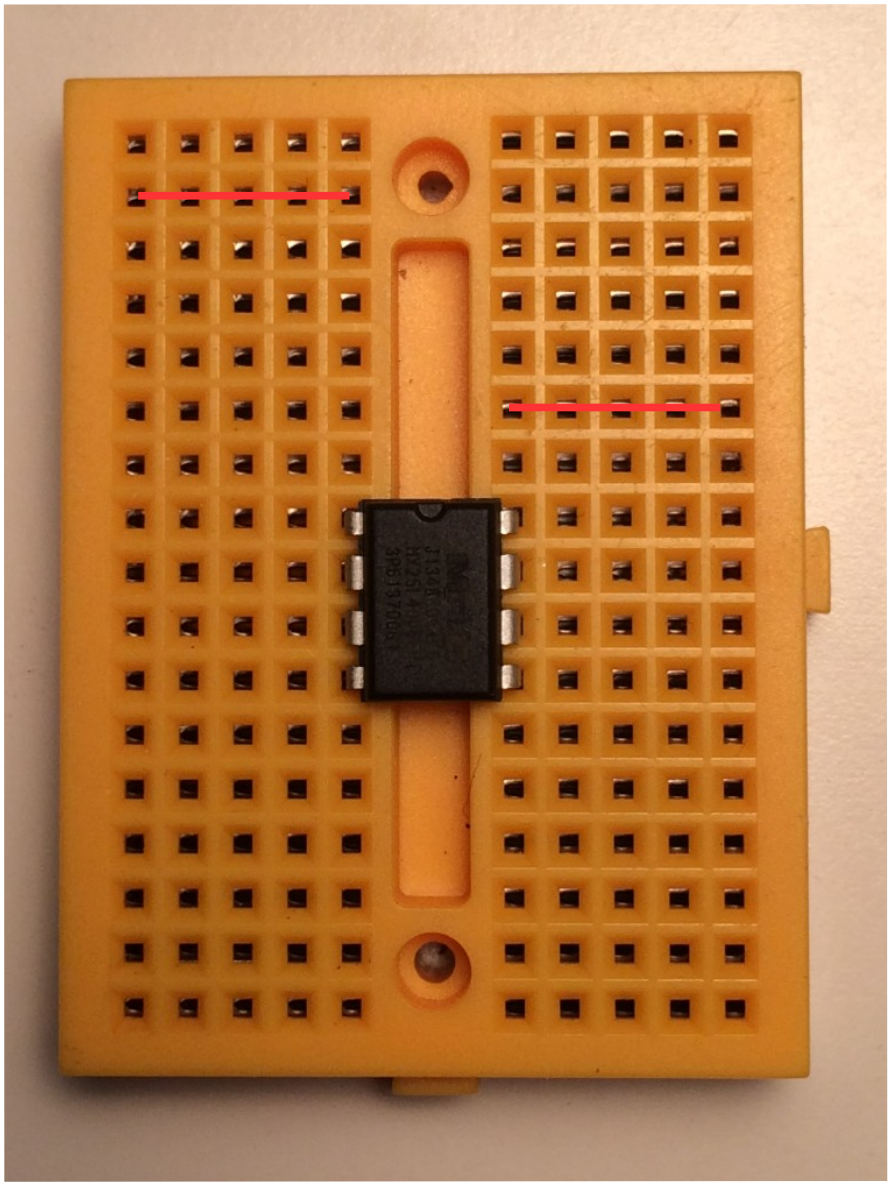
Datasheet in git



Lab 1 – SPI Flash Extraction



Lab 1 – SPI Flash Extraction



Lab 1 – SPI Flash Extraction

GoodFET provides the following SPI Flash functions

- info
- dump $foo.rom [0x$start 0x$stop]
- erase
- flash $foo.rom [0x$start 0x$stop]
- verify $foo.rom [0x$start 0x$stop]
- peek 0x$start [0x$stop]
- poke 0x$adr 0x$val

Lab 1 – SPI Flash Extraction

GoodFET provides the following SPI Flash functions

- info
- dump $foo.rom [0x$start 0x$stop]
- erase
- flash $foo.rom [0x$start 0x$stop]
- verify $foo.rom [0x$start 0x$stop]
- peek 0x$start [0x$stop]
- poke 0x$adr 0x$val

Lab 1 – SPI Flash Extraction

Install GoodFET clients

```
# cd ~

# git clone https://github.com/travisgoodspeed/goodfet
.
.
.
# cd goodfet

# cd client

# make install link

# cd ~
```

Lab 1 – SPI Flash Extraction

MACRONIX
INTERNATIONAL CO., LTD.

MX25L4006E

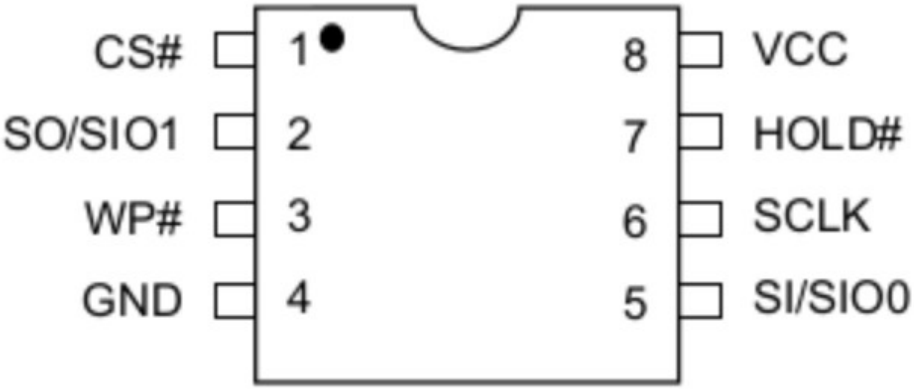
3V, 4M-BIT [x 1/x 2]
CMOS SERIAL FLASH MEMORY

Key Features

- *Hold Feature*
- *Low Power Consumption*
- *Auto Erase and Auto Program Algorithms*
- *Provides sequential read operation on whole chip*

Lab 1 – SPI Flash Extraction

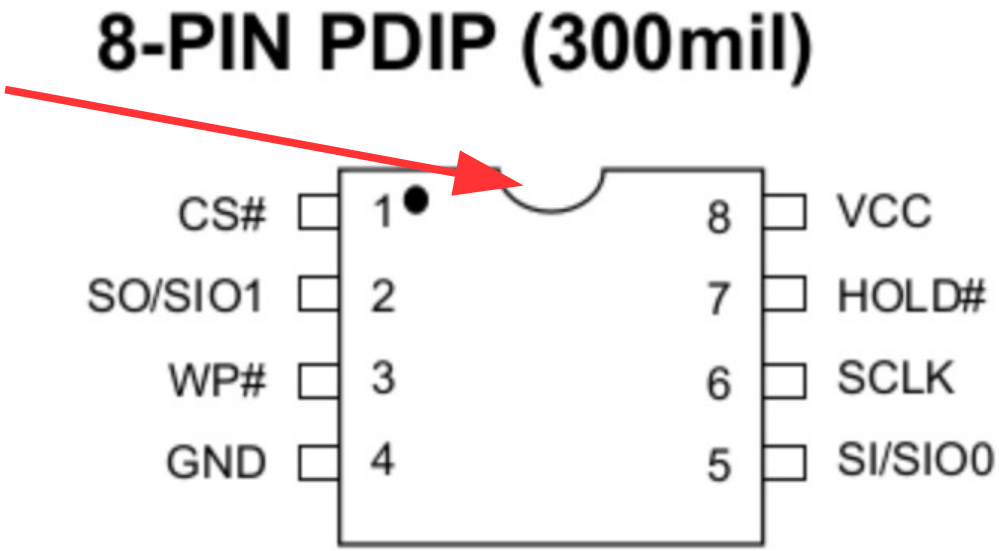
8-PIN PDIP (300mil)



PIN DESCRIPTION

SYMBOL	DESCRIPTION
CS#	Chip Select
SI/SIO0	Serial Data Input (for 1 x I/O) / Serial Data Input & Output (for Dual Output mode)
SO/SIO1	Serial Data Output (for 1 x I/O) / Serial Data Output (for Dual Output mode)
SCLK	Clock Input
WP#	Write Protection
HOLD#	Hold, to pause the device without deselecting the device
VCC	+ 3.3V Power Supply
GND	Ground

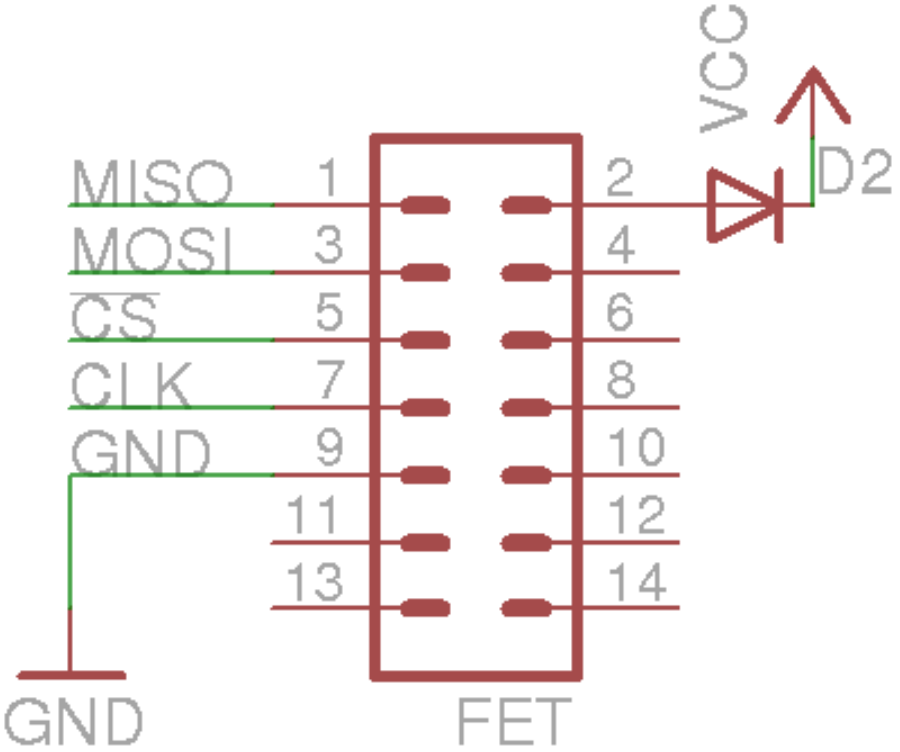
Lab 1 – SPI Flash Extraction

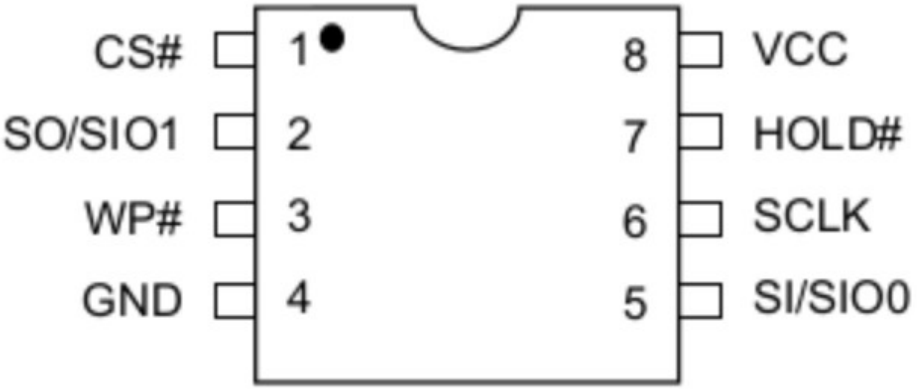


PIN DESCRIPTION

SYMBOL	DESCRIPTION
CS#	Chip Select
SI/SIO0	Serial Data Input (for 1 x I/O) / Serial Data Input & Output (for Dual Output mode)
SO/SIO1	Serial Data Output (for 1 x I/O) / Serial Data Output (for Dual Output mode)
SCLK	Clock Input
WP#	Write Protection
HOLD#	Hold, to pause the device without deselecting the device
VCC	+ 3.3V Power Supply
GND	Ground

Lab 1 – SPI Flash Extraction





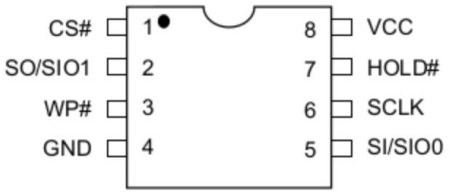
Connect 6 specified pins on GoodFET to SPI flash chip

PIN DESCRIPTION

SYMBOL	DESCRIPTION
CS#	Chip Select
SI/SIO0	Serial Data Input (for 1 x I/O) / Serial Data Input & Output (for Dual Output mode)
SO/SIO1	Serial Data Output (for 1 x I/O) / Serial Data Output (for Dual Output mode)
SCLK	Clock Input
WP#	Write Protection
HOLD#	Hold, to pause the device without deselecting the device
VCC	+ 3.3V Power Supply
GND	Ground

Lab 1 – SPI Flash Extraction

8-PIN PDIP (300mil)



PIN DESCRIPTION

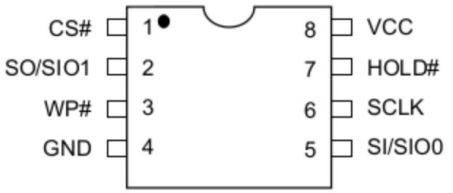
SYMBOL	DESCRIPTION
CS#	Chip Select
SI/SIO0	Serial Data Input (for 1 x I/O) / Serial Data Input & Output (for Dual Output mode)
SO/SIO1	Serial Data Output (for 1 x I/O) / Serial Data Output (for Dual Output mode)
SCLK	Clock Input
WP#	Write Protection
HOLD#	Hold, to pause the device without deselecting the device
VCC	+ 3.3V Power Supply
GND	Ground

Lab 1 – SPI Flash Extraction

Notes

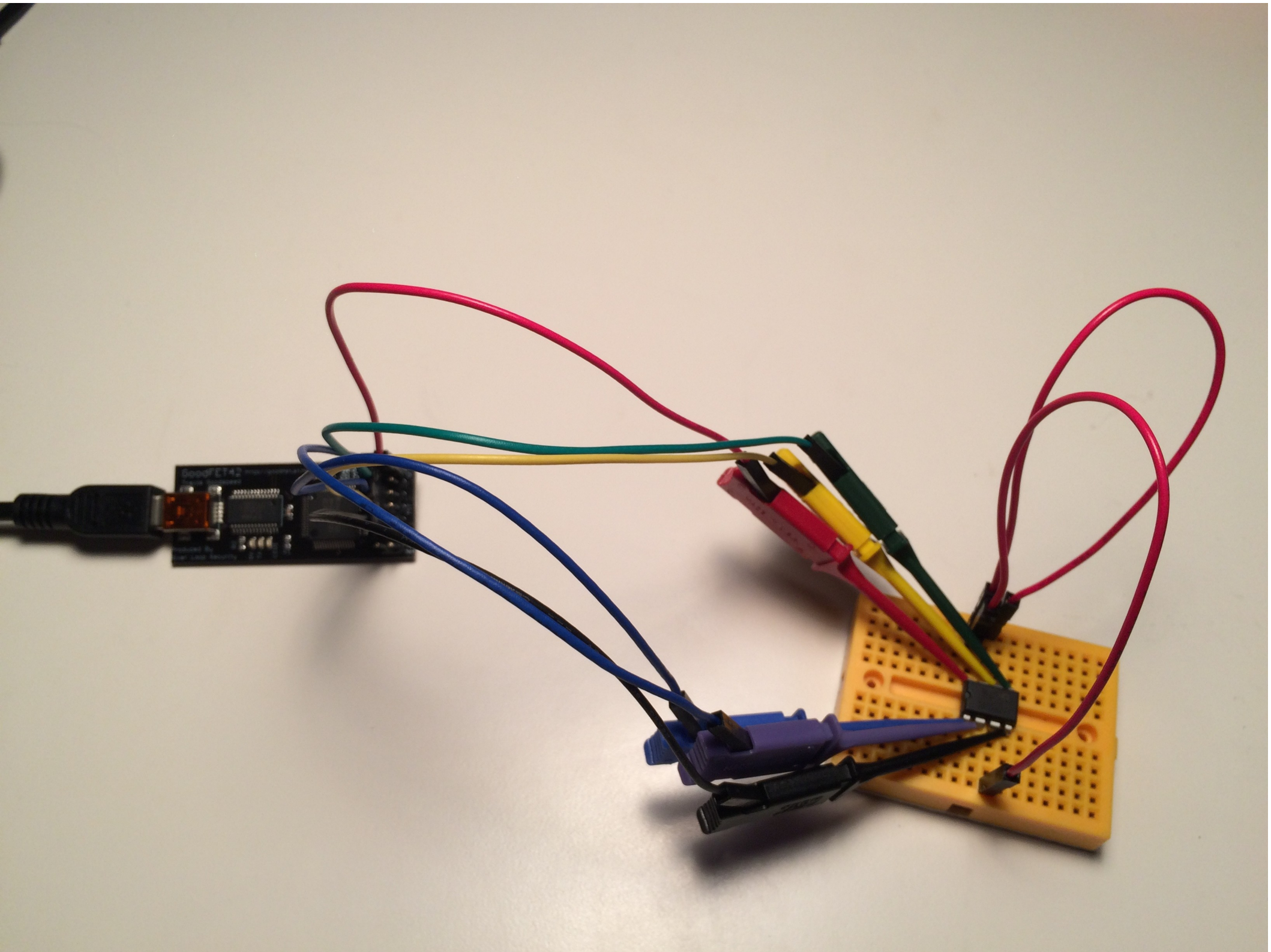
Unlike I2C and SPI EEPROM devices, SPI Flash devices rarely have internal pulling resistors on the !WP and !HOLD pins. You must explicitly pull these up.

8-PIN PDIP (300mil)



PIN DESCRIPTION

SYMBOL	DESCRIPTION
CS#	Chip Select
SI/SIO0	Serial Data Input (for 1 x I/O) / Serial Data Input & Output (for Dual Output mode)
SO/SIO1	Serial Data Output (for 1 x I/O) / Serial Data Output (for Dual Output mode)
SCLK	Clock Input
WP#	Write Protection
HOLD#	Hold, to pause the device without deselecting the device
VCC	+ 3.3V Power Supply
GND	Ground



Lab 1 – SPI Flash Extraction

```
# goodfet.spiflash info

Ident as MXIC MX25L4005
Manufacturer: c2 MXIC
Type: 20
Capacity: 13 (524288 bytes)

# goodfet.spiflash dump spiFlashDump.rom

Dumping code from 000000 to 080000 as spiFlashDump.rom.
.
.
.
# sha1sum spiFlashDump.rom

e1cb0373246c32621a45778c735b9b08078b5f04
```

LUNCH

JTAG

- Boundary Scan
- Programming
- Debugging

- Daisy Chaining

- Manufacturer specific JTAG interfaces

Boundary Scan

- Manually set IC pins high or low
- Read pins from same IC or others

- Simple way to test a circuit

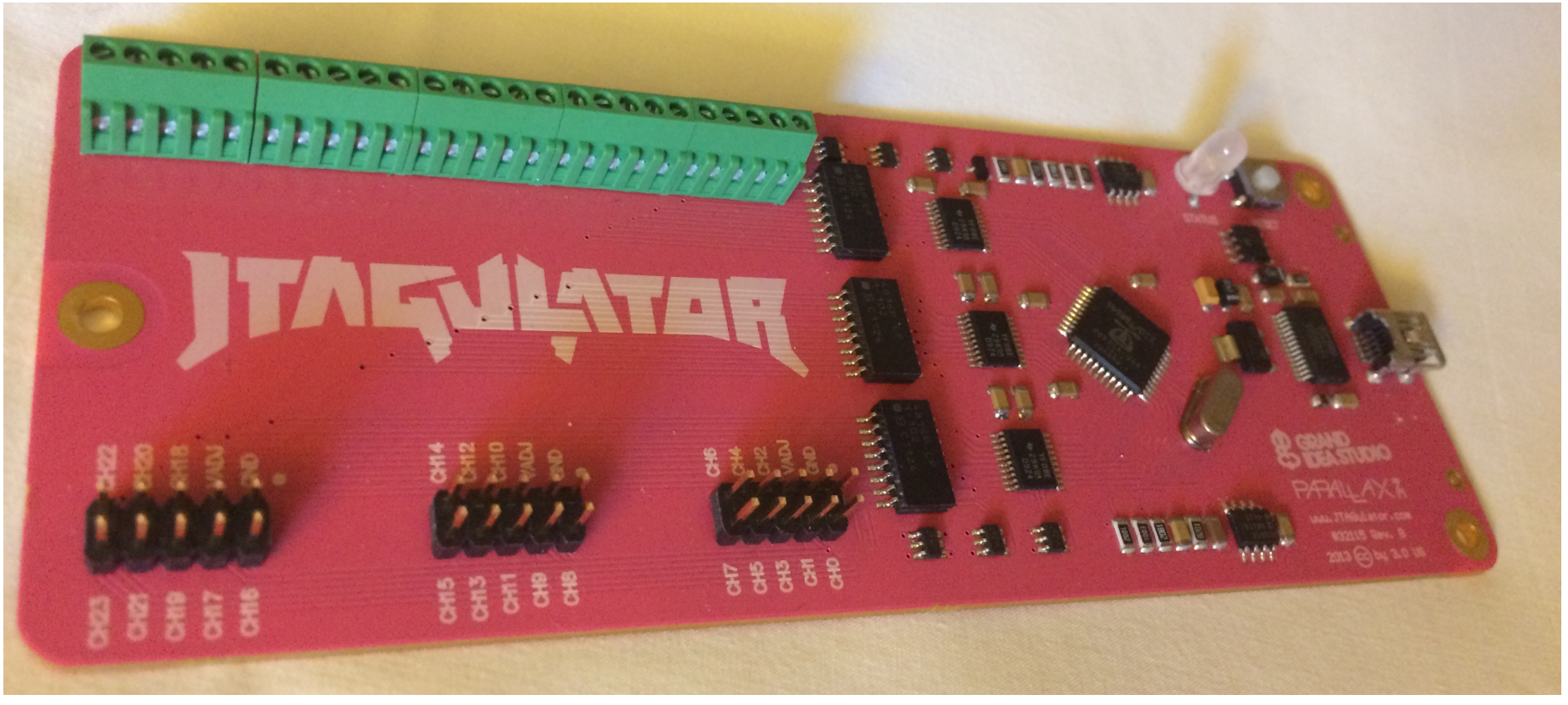
Programming

- Read / write flash memory (Firmware)
- Some standarized commands
- Manufacurer specific extended commands

- Not all manufacturers use JTAG standards for programming
- Some manufacturers allow multiple methods

JTAGulator

- Designed by Joe Grand
- Enumerate pins / test points and discover JTAG layout



JTAG Security

- Read / writes can be password protected
- Fuses can be 'blown' which protects from reading / writing firmware

- Fun project : brute force JTAG passwords

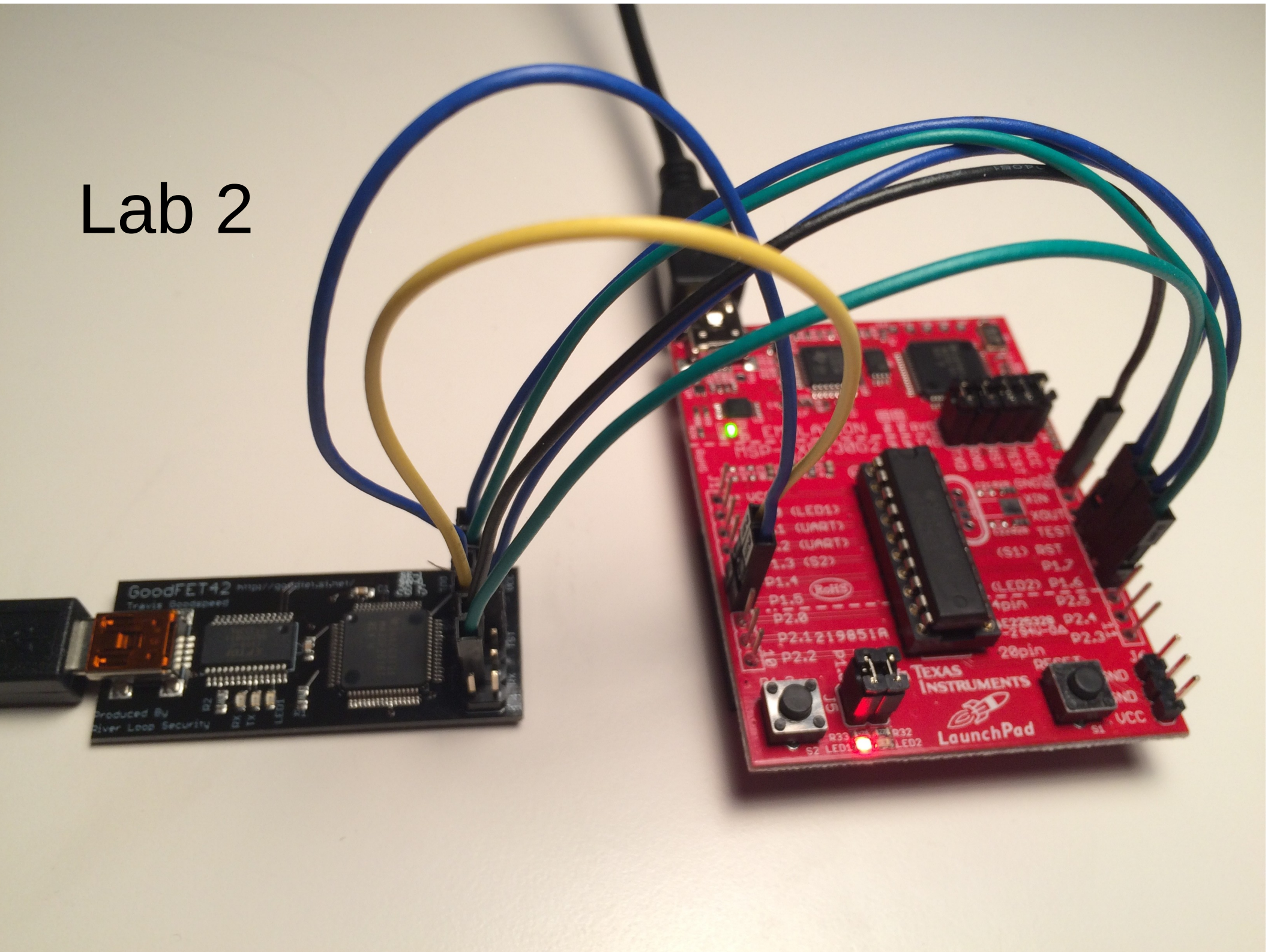
Glitching

- Varying voltage levels
- Potential to trick MCU into allowing single byte / block or full firmware reads / writes

Decapping

- Reset JTAG fuses via laser
- Reset JTAG fuses with light

Lab 2


GoodFET42
Produced By
Diver Loop Security
TEXAS INSTRUMENTS
LaunchPad
P2.1219851A
20pin
VCC

Lab 2 – Firmware Extraction

GoodFET provides the following MSP430 functions

- info
- dump $foo.hex [0x$start 0x$stop]
- erase
- flash $foo.rom [0x$start 0x$stop]
- verify $foo.rom [0x$start 0x$stop]
- peek 0x$start [0x$stop]
- poke 0x$adr 0x$val
- run

Lab 2 – Firmware Extraction

GoodFET provides the following MSP430 functions

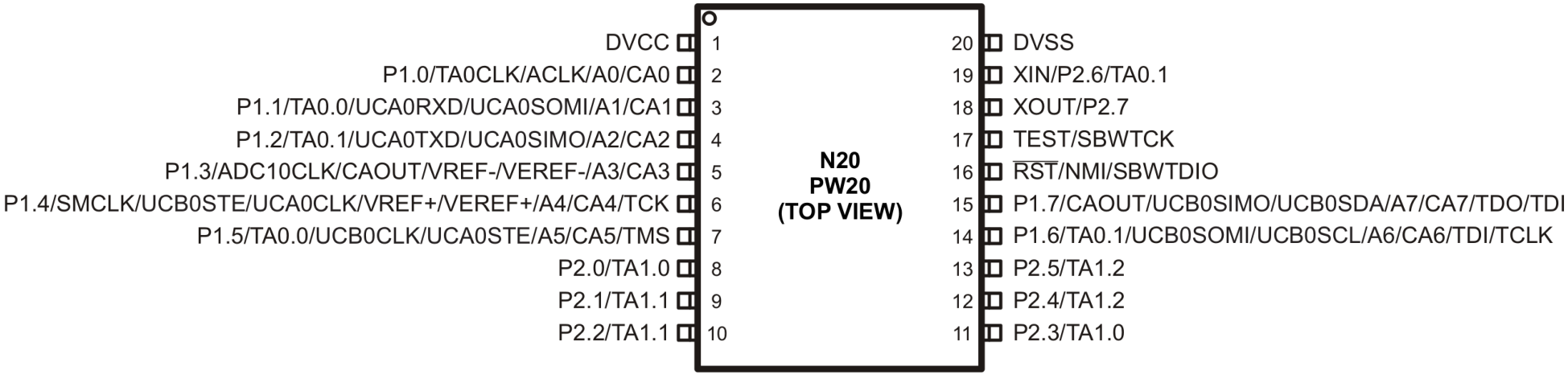
- info
- dump $foo.hex [0x$start 0x$stop]
- erase
- flash $foo.rom [0x$start 0x$stop]
- verify $foo.rom [0x$start 0x$stop]
- peek 0x$start [0x$stop]
- poke 0x$adr 0x$val
- run

Pinout

Name	Pin		Name
TDO	1	2	Vcc
TDI	3	4	Vcc
TMS	5	6	
TCK	7	8	TEST
GND	9	10	
RST	11	12	
	13	14	

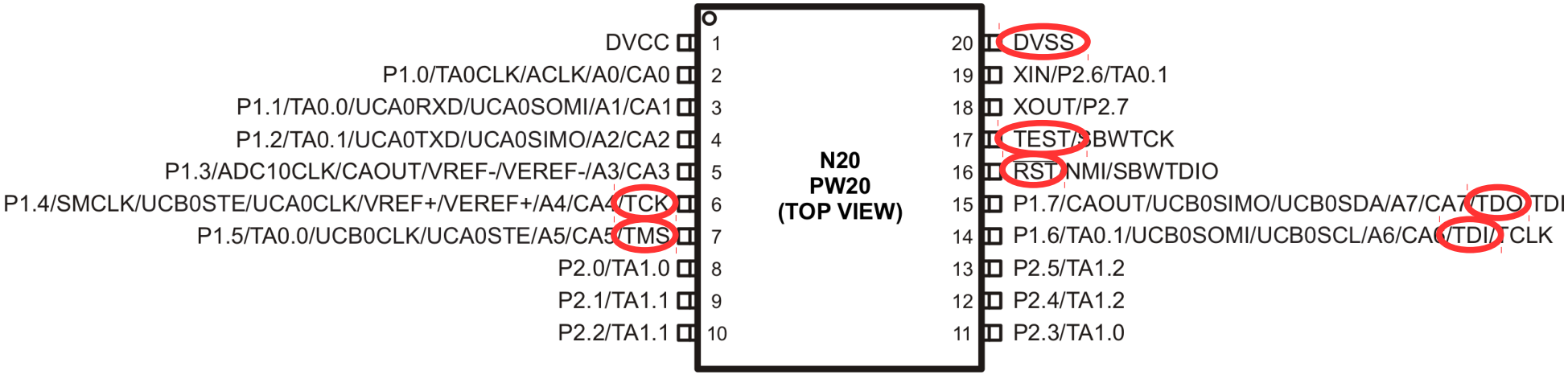
Pinout

Name	Pin		Name
TDO	1	2	Vcc
TDI	3	4	Vcc
TMS	5	6	
TCK	7	8	TEST
GND	9	10	
RST	11	12	
	13	14	



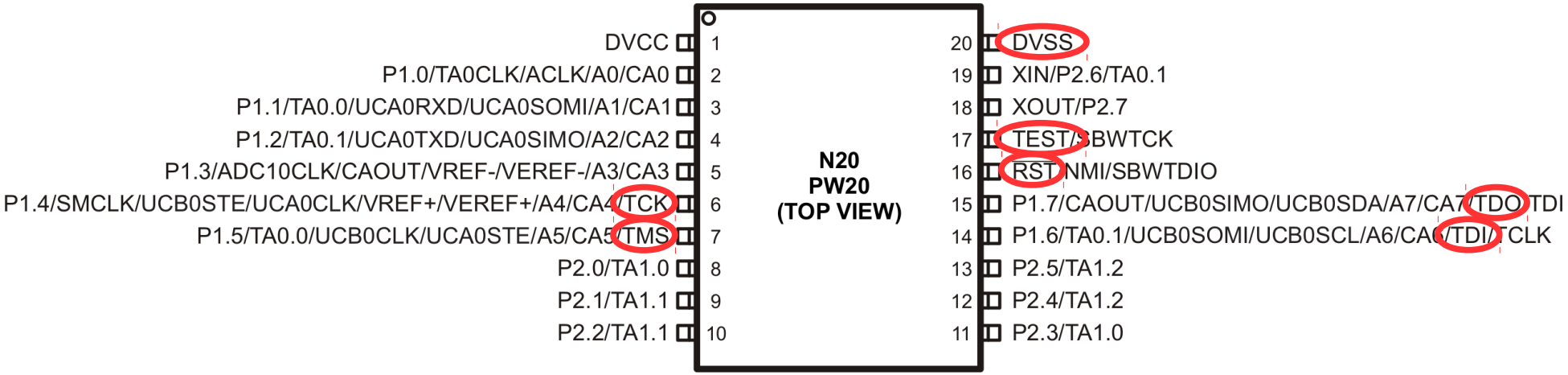
Pinout

Name	Pin		Name
TDO	1	2	Vcc
TDI	3	4	Vcc
TMS	5	6	
TCK	7	8	TEST
GND	9	10	
RST	11	12	
	13	14	

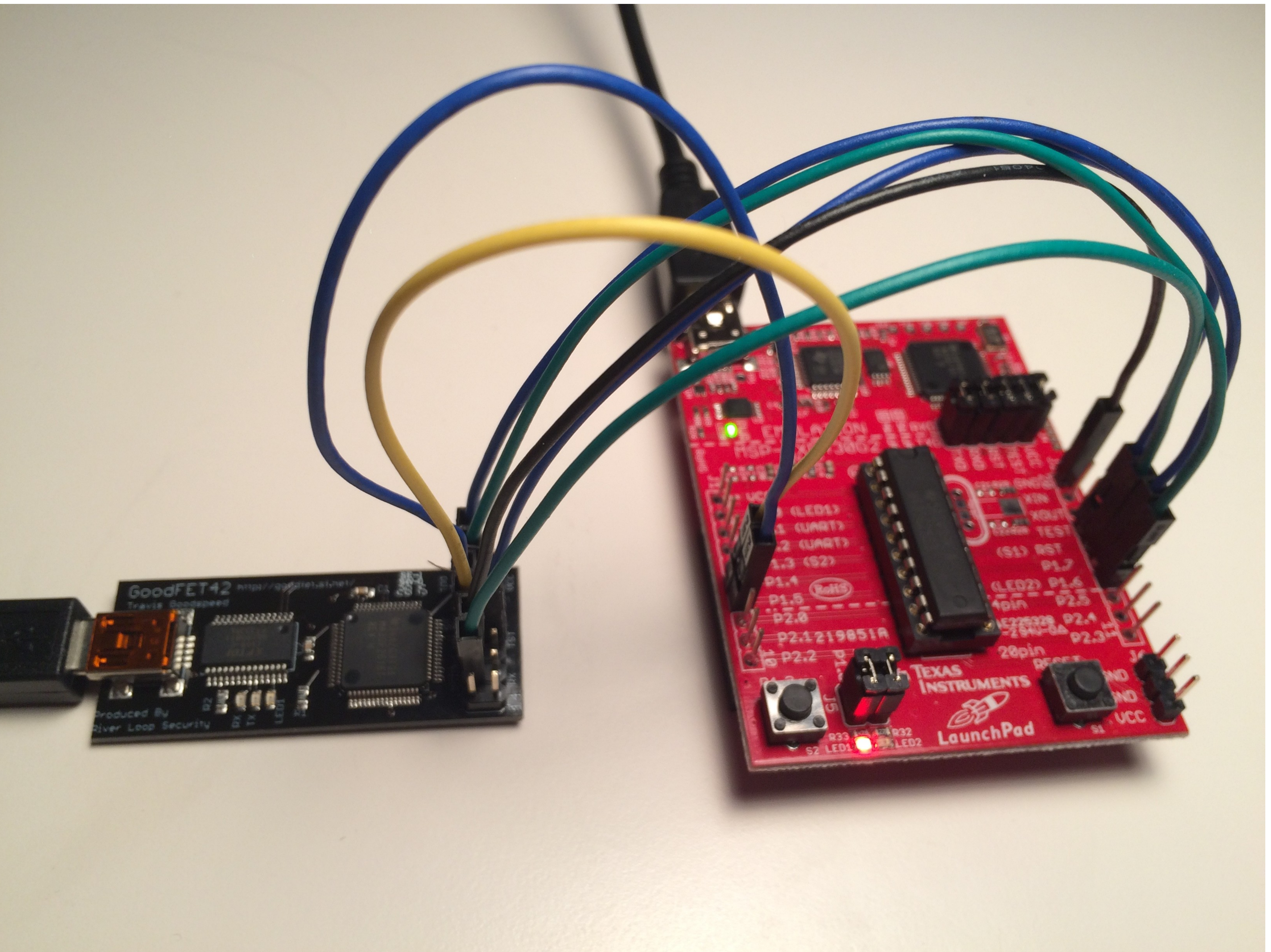


Pinout

Name	Pin		Name
TDO	1	2	Vcc
TDI	3	4	Vcc
TMS	5	6	
TCK	7	8	TEST
GND	9	10	
RST	11	12	
	13	14	




```
$ goodfet.msp430 info
Identifies as MSP430G2553 (2553)

$ goodfet.msp430 dump msp430.hex
.
.
.
```


GoodFET42
Produced By
River Loop Security
TEXAS INSTRUMENTS
LaunchPad
RESET
20pin
P1.4
P1.5
P2.0
P2.2
P1.7
P1.6
P2.5
P2.4
P2.3
(S1) RST
TEST
XOUT
XIN
GND
VCC

Dump the file multiple times – **sha1sum** is different. Why?

Dump the file multiple times – **sha1sum** is different. Why?

Table 8. Memory Organization

		MSP430G2153	MSP430G2253 MSP430G2213	MSP430G2353 MSP430G2313	MSP430G2453 MSP430G2413	MSP430G2553 MSP430G2513
Memory	Size	1kB	2kB	4kB	8kB	16kB
Main: interrupt vector	Flash	0xFFFF to 0xFFC0	0xFFFF to 0xFFC0	0xFFFF to 0xFFC0	0xFFFF to 0xFFC0	0xFFFF to 0xFFC0
Main: code memory	Flash	0xFFFF to 0xFC00	0xFFFF to 0xF800	0xFFFF to 0xF000	0xFFFF to 0xE000	0xFFFF to 0xC000
Information memory	Size	256 Byte	256 Byte	256 Byte	256 Byte	256 Byte
	Flash	010FFh to 01000h	010FFh to 01000h	010FFh to 01000h	010FFh to 01000h	010FFh to 01000h
RAM	Size	256 Byte	256 Byte	256 Byte	512 Byte	512 Byte
		0x02FF to 0x0200	0x02FF to 0x0200	0x02FF to 0x0200	0x03FF to 0x0200	0x03FF to 0x0200
Peripherals	16-bit	01FFh to 0100h	01FFh to 0100h	01FFh to 0100h	01FFh to 0100h	01FFh to 0100h
	8-bit	0FFh to 010h	0FFh to 010h	0FFh to 010h	0FFh to 010h	0FFh to 010h
	8-bit SFR	0Fh to 00h	0Fh to 00h	0Fh to 00h	0Fh to 00h	0Fh to 00h

Dump the file multiple times – **sha1sum** is different. Why?

Table 8. Memory Organization

		MSP430G2153	MSP430G2253 MSP430G2213	MSP430G2353 MSP430G2313	MSP430G2453 MSP430G2413	MSP430G2553 MSP430G2513
Memory	Size	1kB	2kB	4kB	8kB	16kB
Main: interrupt vector	Flash	0xFFFF to 0xFFC0	0xFFFF to 0xFFC0	0xFFFF to 0xFFC0	0xFFFF to 0xFFC0	0xFFFF to 0xFFC0
Main: code memory	Flash	0xFFFF to 0xFC00	0xFFFF to 0xF800	0xFFFF to 0xF000	0xFFFF to 0xE000	0xFFFF to 0xC000
Information memory	Size	256 Byte	256 Byte	256 Byte	256 Byte	256 Byte
	Flash	010FFh to 01000h	010FFh to 01000h	010FFh to 01000h	010FFh to 01000h	010FFh to 01000h
RAM	Size	256 Byte	256 Byte	256 Byte	512 Byte	512 Byte
		0x02FF to 0x0200	0x02FF to 0x0200	0x02FF to 0x0200	0x03FF to 0x0200	0x03FF to 0x0200
Peripherals	16-bit	01FFh to 0100h	01FFh to 0100h	01FFh to 0100h	01FFh to 0100h	01FFh to 0100h
	8-bit	0FFh to 010h	0FFh to 010h	0FFh to 010h	0FFh to 010h	0FFh to 010h
	8-bit SFR	0Fh to 00h	0Fh to 00h	0Fh to 00h	0Fh to 00h	0Fh to 00h

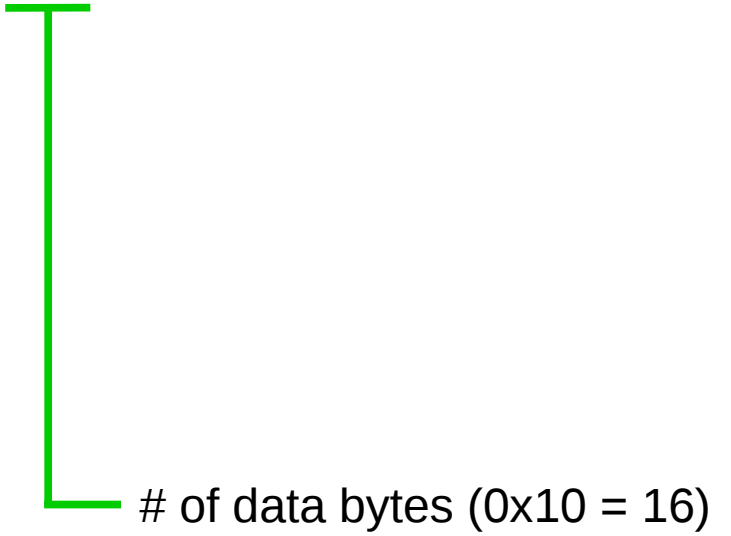
```
$ goodfet.msp430 dump msp430code.hex 0xC000 0xFFFF
.
.

$ sha1sum msp430code.hex

4444abe1a86dab7df4358ad86a7ad289212b1a59
```

Hex files (They're in hex!)

:10C00000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF40

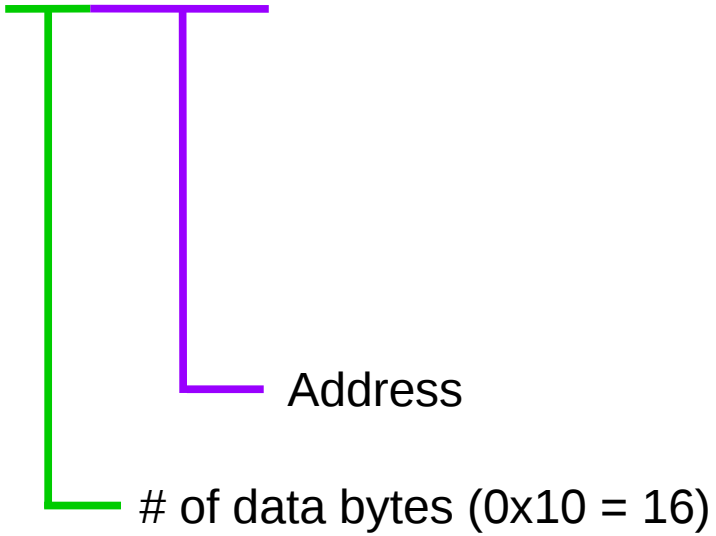
Hex files (They're in hex!)

:10C00000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF40



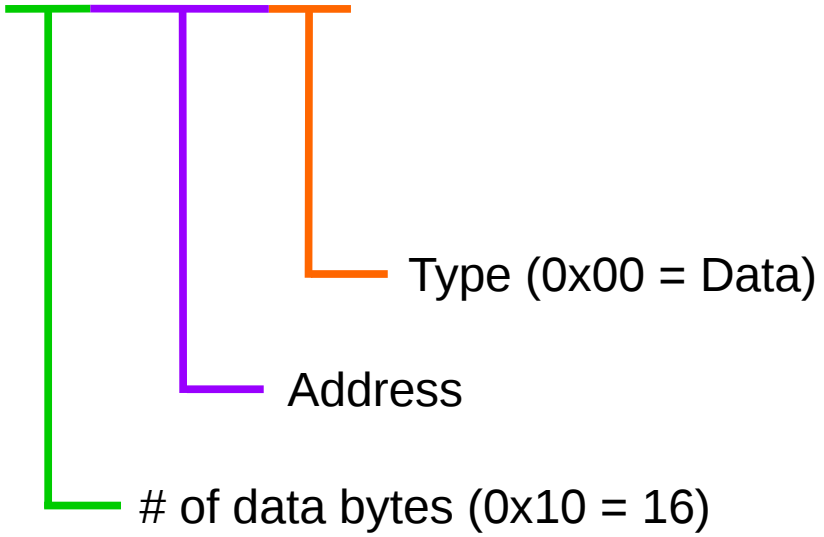
Hex files (They're in hex!)

:10C00000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF40

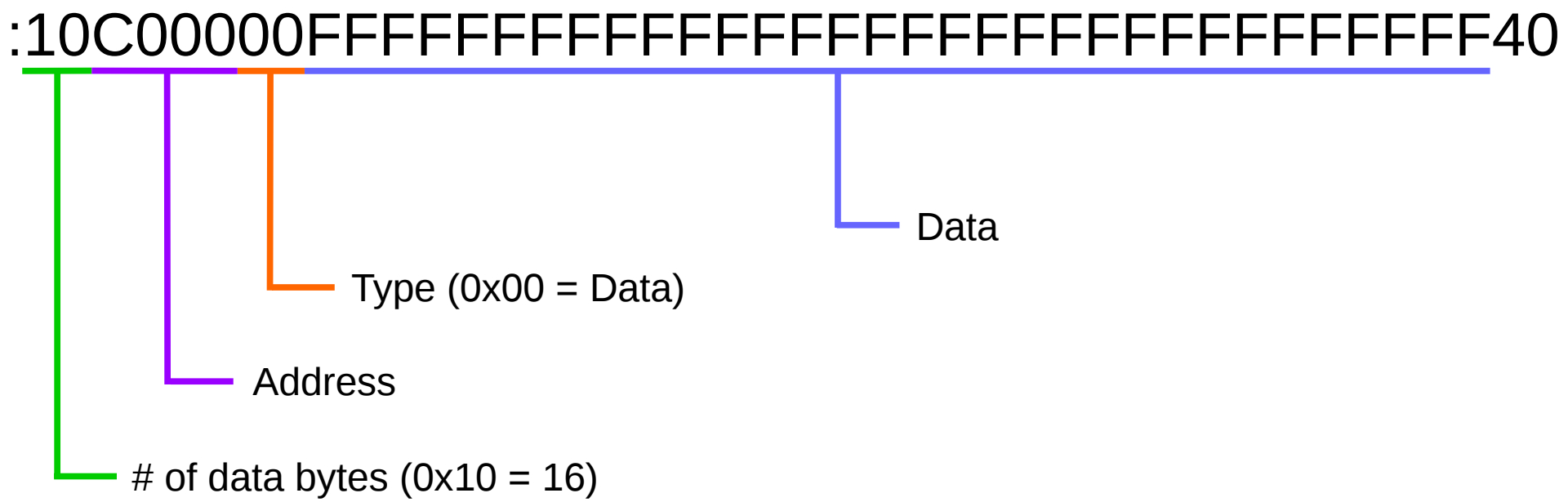


Hex files (They're in hex!)

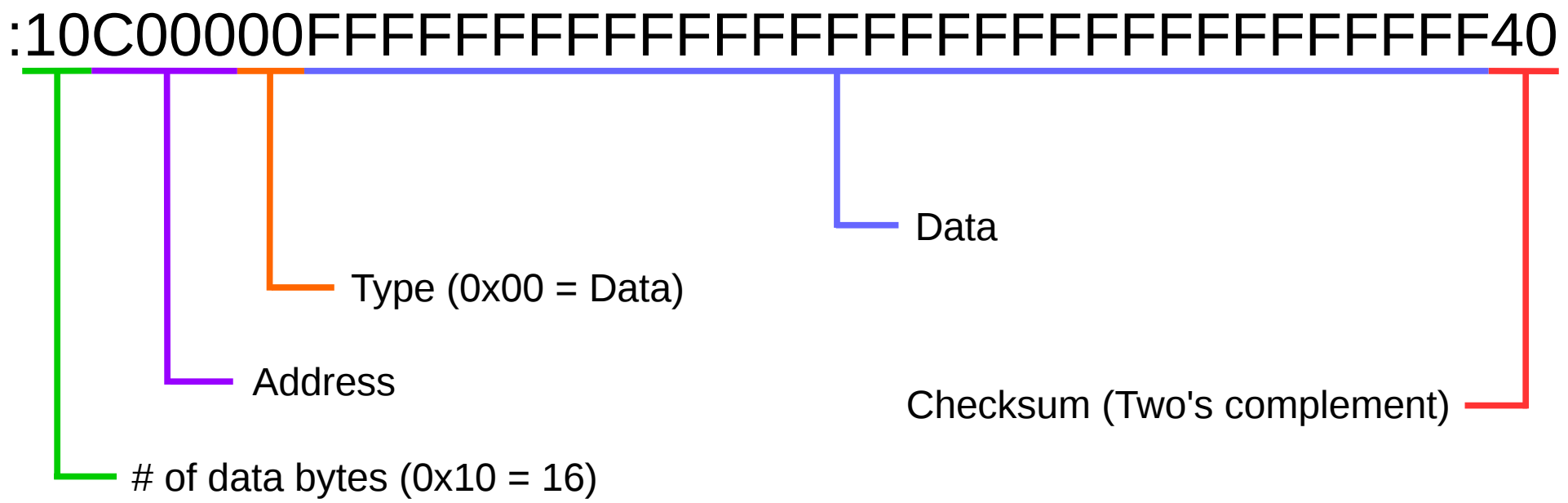
:10C00000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF40



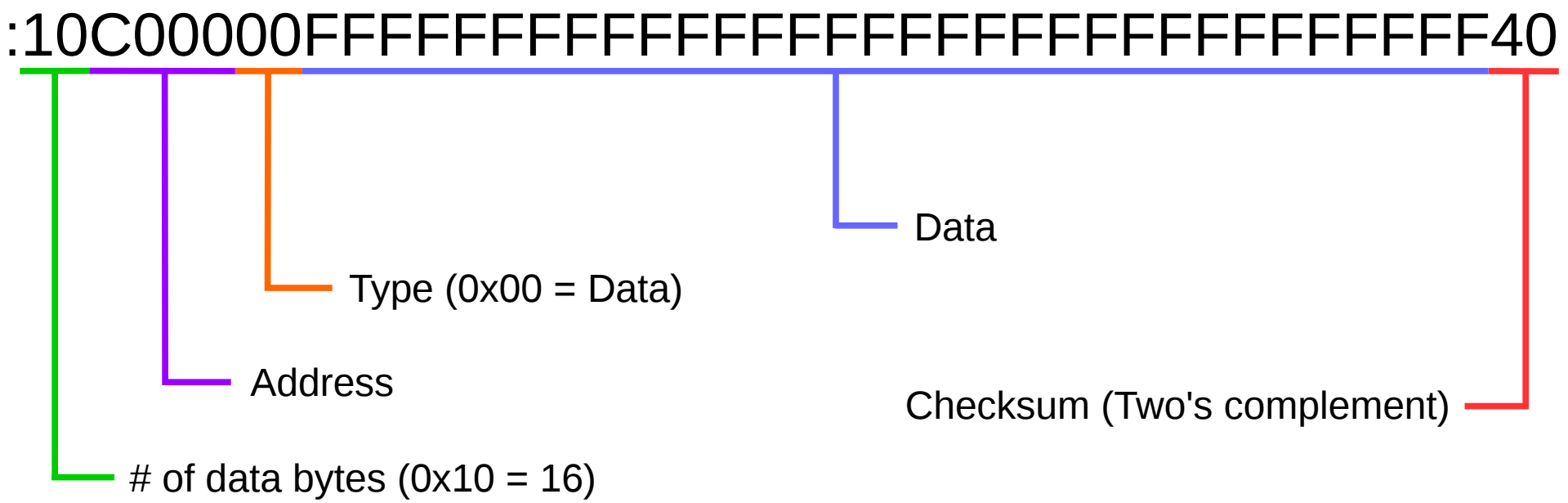
Hex files (They're in hex!)



Hex files (They're in hex!)



Hex files (They're in hex!)



:10F80000B240805A2001D242FF105700D242FE106F

hex2bin

https://sourceforge.net/projects/hex2bin

```
$ tar -jxvf Hex2bin-2.2.tar.bz2
```

hex2bin

https://sourceforge.net/projects/hex2bin

```
# tar -jxvf Hex2bin-2.2.tar.bz2
```

```
# cd Hex2bin-2.2

# ./hex2bin -a 0x0 -m msp430 -D msp430code.hex
.
.
```

Firmware Analysis

Which output port and pins drive the blinking LEDs?

Firmware Analysis

Table 15. Peripherals With Byte Access (continued)

MODULE	REGISTER DESCRIPTION	REGISTER NAME	OFFSET
Port P1	Port P1 selection 2	P1SEL2	041h
	Port P1 resistor enable	P1REN	027h
	Port P1 selection	P1SEL	026h
	Port P1 interrupt enable	P1IE	025h
	Port P1 interrupt edge select	P1IES	024h
	Port P1 interrupt flag	P1IFG	023h
	Port P1 direction	P1DIR	022h
	Port P1 output	P1OUT	021h
	Port P1 input	P1IN	020h
Special Function	SFR interrupt flag 2	IFG2	003h
	SFR interrupt flag 1	IFG1	002h
	SFR interrupt enable 2	IE2	001h
	SFR interrupt enable 1	IE1	000h

Firmware Analysis

Table 15. Peripherals With Byte Access (continued)

MODULE	REGISTER DESCRIPTION	REGISTER NAME	OFFSET
Port P1	Port P1 selection 2	P1SEL2	041h
	Port P1 resistor enable	P1REN	027h
	Port P1 selection	P1SEL	026h
	Port P1 interrupt enable	P1IE	025h
	Port P1 interrupt edge select	P1IES	024h
	Port P1 interrupt flag	P1IFG	023h
	Port P1 direction	P1DIR	022h
	Port P1 output	P1OUT	021h
	Port P1 input	P1IN	020h
Special Function	SFR interrupt flag 2	IFG2	003h
	SFR interrupt flag 1	IFG1	002h
	SFR interrupt enable 2	IE2	001h
	SFR interrupt enable 1	IE1	000h

Firmware Analysis

Table 16. Port P1 (P1.0 to P1.2) Pin Functions

PIN NAME (P1.x)	x	FUNCTION	CONTROL BITS AND SIGNALS[1]				
			P1DIR.x	P1SEL.x	P1SEL2.x	ADC10AE.x INCH.x=1[2]	CAPD.y
P1.0/	0	P1.x (I/O)	I: 0; O: 1	0	0	0	0
TA0CLK/		TA0.TACLK	0	1	0	0	0
ACLK/		ACLK	1	1	0	0	0
A0[2]/		A0	X	X	X	1 (y = 0)	0
CA0/		CA0	X	X	X	0	1 (y = 0)
Pin Osc		Capacitive sensing	X	0	1	0	0

Firmware Analysis

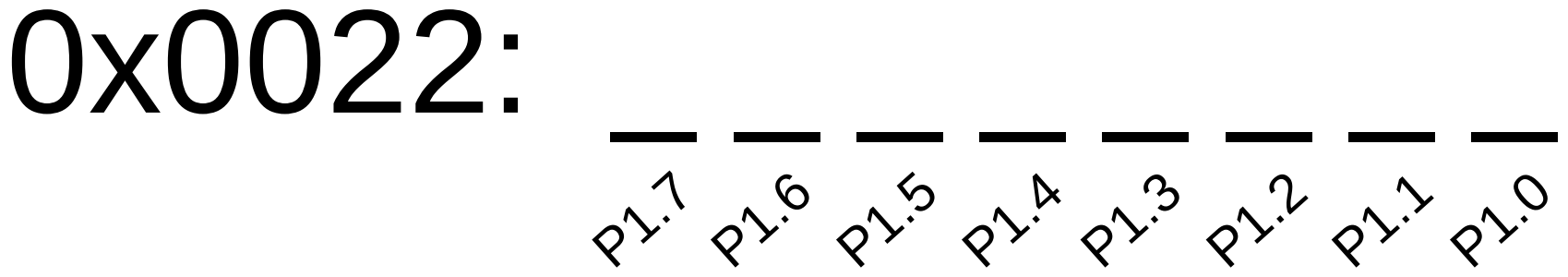
Table 16. Port P1 (P1.0 to P1.2) Pin Functions

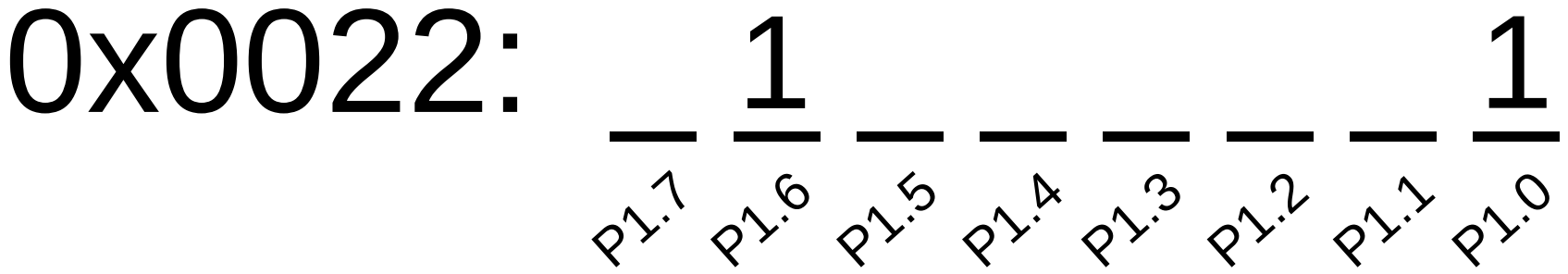
PIN NAME (P1.x)	x	FUNCTION	CONTROL BITS AND SIGNALS[1]				
			P1DIR.x	P1SEL.x	P1SEL2.x	ADC10AE.x INCH.x=1[2]	CAPD.y
P1.0/	0	P1.x (I/O)	I: 0; O: 1	0	0	0	0
TA0CLK/		TA0.TACLK	0	1	0	0	0
ACLK/		ACLK	1	1	0	0	0
A0[2]/		A0	X	X	X	1 (y = 0)	0
CA0/		CA0	X	X	X	0	1 (y = 0)
Pin Osc		Capacitive sensing	X	0	1	0	0

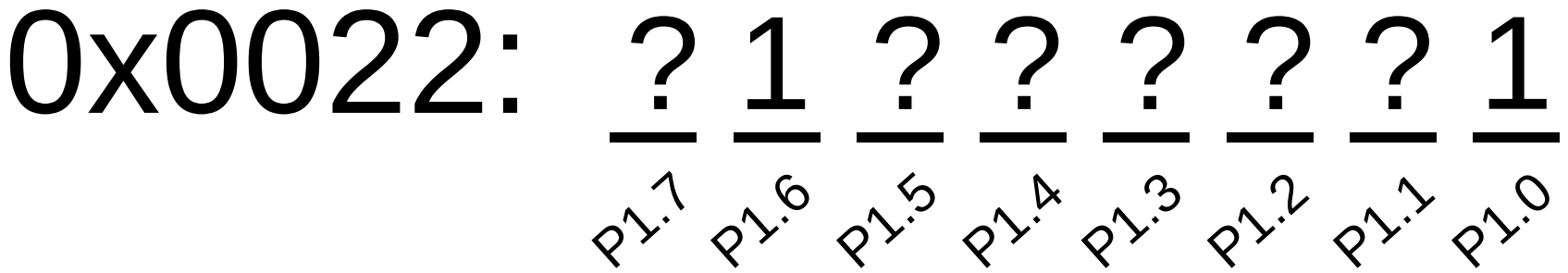
Firmware Analysis

Table 16. Port P1 (P1.0 to P1.2) Pin Functions

PIN NAME (P1.x)	x	FUNCTION	CONTROL BITS AND SIGNALS[1]				
			P1DIR.x	P1SEL.x	P1SEL2.x	ADC10AE.x INCH.x=1[2]	CAPD.y
P1.0/	0	P1.x (I/O)	I: 0; O: 1	0	0	0	0
TA0CLK/		TA0.TACLK	0	1	0	0	0
ACLK/		ACLK	1	1	0	0	0
A0[2]/		A0	X	X	X	1 (y = 0)	0
CA0/		CA0	X	X	X	0	1 (y = 0)
Pin Osc		Capacitive sensing	X	0	1	0	0
P1.6/	6	P1.x (I/O)	I: 0; O: 1	0	0	0	0	0
TA0.1/		TA0.1	1	1	0	0	0	0
UCB0SOMI/		UCB0SOMI	from USCI	1	1	0	0	0
UCB0SCL/		UCB0SCL	from USCI	1	1	0	0	0
A6[2]/		A6	X	X	X	1 (y = 6)	0	0
CA6		CA6	X	X	X	0	0	1 (y = 6)
TDI/TCLK/		TDI/TCLK	X	X	X	0	1	0
Pin Osc		Capacitive sensing	X	0	1	0	0	0


0x0022:
P1.7
P1.6
P1.5
P1.4
P1.3
P1.2
P1.1
P1.0


0x0022:
1
1
P1.7
P1.6
P1.5
P1.4
P1.3
P1.2
P1.1
P1.0


0x0022: ? 1 ? ? ? ? ? 1
P1.7
P1.6
P1.5
P1.4
P1.3
P1.2
P1.1
P1.0

0x0022: ? 1 ? ? ? ? ? 1

P1.7 P1.6 P1.5 P1.4 P1.3 P1.2 P1.1 P1.0

x & 65 == 65

0x0022: ? 1 ? ? ? ? ? 1

P1.7 P1.6 P1.5 P1.4 P1.3 P1.2 P1.1 P1.0

x & 65 == 65

x & 0x41 == 0x41

Radare2

```
# r2 -a msp430 msp430code.bin
[0x00000000]> s 0xf800
[0x0000f800]> aa
[0x0000f800]>
```

V – enter visual mode
p – switch view to disassembly

```
[0x0000f800 96% 115 msp430code.bin]> pd $r @ fcn.0000f800
/ (fcn) fcn.0000f800 168
|           0x0000f800    b240805a2001   mov #0x5a80, &0x0120
|           0x0000f806    d242ff105700   mov.b &0x10ff, &0x0057
|           0x0000f80c    d242fe105600   mov.b &0x10fe, &0x0056
|           0x0000f812    f2f0f9005800   and.b #0x00f9, &0x0058
|           0x0000f818    f2c22200       dint
|           0x0000f81c    f2d22100       eint
|           0x0000f820    f2d22700       eint
|           0x0000f824    f2d22400       eint
|           0x0000f828    f2c22300       dint
|           0x0000f82c    f2d22500       eint
|           0x0000f830    f2d041002200   bis.b #0x0041, &0x0022
|           0x0000f836    f2f0be002100   and.b #0x00be, &0x0021
|           0x0000f83c    b0124efc       call #0xfc4e                ;[0]
|             ^- unk() ; fcn.0000f800+-63489
|           0x0000f840    d2430102       mov.b #1, &0x0201
|           0x0000f844    b012a0fa       call #0xfaa0                ;[0]
|             ^- unk() ; fcn.0000f800+-63489
|           0x0000f848    b012c4f9       call #0xf9c4                ;[0]
|             ^- unk() ; fcn.0000f800+-63489
|           0x0000f84c    32d2           eint
|           0x0000f84e    3b40e803       mov #0x03e8, r11
```

```
[0x0000f800 96% 115 msp430code.bin]> pd $r @ fcn.0000f800
/ (fcn) fcn.0000f800 168
|           0x0000f800    b240805a2001   mov #0x5a80, &0x0120
|           0x0000f806    d242ff105700   mov.b &0x10ff, &0x0057
|           0x0000f80c    d242fe105600   mov.b &0x10fe, &0x0056
|           0x0000f812    f2f0f9005800   and.b #0x00f9, &0x0058
|           0x0000f818    f2c22200       dint
|           0x0000f81c    f2d22100       eint
|           0x0000f820    f2d22700       eint
|           0x0000f824    f2d22400       eint
|           0x0000f828    f2c22300       dint
|           0x0000f82c    f2d22500       eint
|           0x0000f830    f2d041002200   bis.b #0x0041, &0x0022
|           0x0000f836    f2f0be002100   and.b #0x00be, &0x0021
|           0x0000f83c    b0124efc       call #0xfc4e                ;[0]
|             ^- unk() ; fcn.0000f800+-63489
|           0x0000f840    d2430102       mov.b #1, &0x0201
|           0x0000f844    b012a0fa       call #0xfaa0                ;[0]
|             ^- unk() ; fcn.0000f800+-63489
|           0x0000f848    b012c4f9       call #0xf9c4                ;[0]
|             ^- unk() ; fcn.0000f800+-63489
|           0x0000f84c    32d2           eint
|           0x0000f84e    3b40e803       mov #0x03e8, r11
```

← Up to date r2 is better (from git)

binutils-msp430

```
# apt-get install binutils-msp430
.
.
.
# msp430-objdump -m msp430 -D msp430code.hex [check this]
.
.
.
# msp430-objdump -m msp430 -D msp430code.hex > msp430objdump.txt
```

```
f800:   b2 40 80 5a     mov #23168, &0x0120 ;#0x5a80
f804:   20 01
f806:   d2 42 ff 10     mov.b   &0x10ff,&0x0057
f80a:   57 00
f80c:   d2 42 fe 10     mov.b   &0x10fe,&0x0056
f810:   56 00
f812:   f2 f0 f9 00     and.b   #249,   &0x0058 ;#0x00f9
f816:   58 00
f818:   f2 c2 22 00     bic.b   #8, &0x0022 ;r2 As==11
f81c:   f2 d2 21 00     bis.b   #8, &0x0021 ;r2 As==11
f820:   f2 d2 27 00     bis.b   #8, &0x0027 ;r2 As==11
f824:   f2 d2 24 00     bis.b   #8, &0x0024 ;r2 As==11
f828:   f2 c2 23 00     bic.b   #8, &0x0023 ;r2 As==11
f82c:   f2 d2 25 00     bis.b   #8, &0x0025 ;r2 As==11
f830:   f2 d0 41 00     bis.b   #65,    &0x0022 ;#0x0041
f834:   22 00
f836:   f2 f0 be 00     and.b   #190,   &0x0021 ;#0x00be
f83a:   21 00
f83c:   b0 12 4e fc     call    #0xfc4e
f840:   d2 43 01 02     mov.b   #1, &0x0201 ;r3 As==01
f844:   b0 12 a0 fa     call    #0xfaa0
f848:   b0 12 c4 f9     call    #0xf9c4
f84c:   32 d2           eint
f84e:   3b 40 e8 03     mov #1000,  r11 ;#0x03e8
f852:   0b 3c           jmp $+24        ;abs 0xf86a
f854:   c2 43 05 02     mov.b   #0, &0x0205 ;r3 As==00
f858:   b2 f0 ef ff     and #-17,   &0x0162 ;#0xffef
f85c:   62 01
f85e:   b2 f0 ef ff     and #-17,   &0x0164 ;#0xffef
```

```
f800:   b2 40 80 5a     mov #23168, &0x0120 ;#0x5a80
f804:   20 01
f806:   d2 42 ff 10     mov.b   &0x10ff,&0x0057
f80a:   57 00
f80c:   d2 42 fe 10     mov.b   &0x10fe,&0x0056
f810:   56 00
f812:   f2 f0 f9 00     and.b   #249,   &0x0058 ;#0x00f9
f816:   58 00
f818:   f2 c2 22 00     bic.b   #8, &0x0022 ;r2 As==11
f81c:   f2 d2 21 00     bis.b   #8, &0x0021 ;r2 As==11
f820:   f2 d2 27 00     bis.b   #8, &0x0027 ;r2 As==11
f824:   f2 d2 24 00     bis.b   #8, &0x0024 ;r2 As==11
f828:   f2 c2 23 00     bic.b   #8, &0x0023 ;r2 As==11
f82c:   f2 d2 25 00     bis.b   #8, &0x0025 ;r2 As==11
f830:   f2 d0 41 00     bis.b   #65,    &0x0022 ;#0x0041
f834:   22 00
f836:   f2 f0 be 00     and.b   #190,   &0x0021 ;#0x00be
f83a:   21 00
f83c:   b0 12 4e fc     call    #0xfc4e
f840:   d2 43 01 02     mov.b   #1, &0x0201 ;r3 As==01
f844:   b0 12 a0 fa     call    #0xfaa0
f848:   b0 12 c4 f9     call    #0xf9c4
f84c:   32 d2           eint
f84e:   3b 40 e8 03     mov #1000,  r11 ;#0x03e8
f852:   0b 3c           jmp $+24        ;abs 0xf86a
f854:   c2 43 05 02     mov.b   #0, &0x0205 ;r3 As==00
f858:   b2 f0 ef ff     and #-17,   &0x0162 ;#0xffef
f85c:   62 01
f85e:   b2 f0 ef ff     and #-17,   &0x0164 ;#0xffef
```

```
f9c4:   c2 43 02 02     mov.b   #0, &0x0202 ;r3 As==00
f9c8:   b2 40 d0 07     mov #2000,  &0x0172 ;#0x07d0
f9cc:   72 01
f9ce:   b2 40 10 02     mov #528,   &0x0160 ;#0x0210
f9d2:   60 01
f9d4:   b2 40 10 00     mov #16,    &0x0162 ;#0x0010
f9d8:   62 01
f9da:   b2 40 70 00     mov #112,   &0x0164 ;#0x0070
f9de:   64 01
f9e0:   92 43 74 01     mov #1, &0x0174 ;r3 As==01
f9e4:   30 41           ret
f9e6:   d2 93 02 02     cmp.b   #1, &0x0202 ;r3 As==01
f9ea:   0e 24           jz  $+30        ;abs 0xfa08
f9ec:   d2 93 05 02     cmp.b   #1, &0x0205 ;r3 As==01
f9f0:   02 20           jnz $+6         ;abs 0xf9f6
f9f2:   d2 d3 21 00     bis.b   #1, &0x0021 ;r3 As==01
f9f6:   e2 93 05 02     cmp.b   #2, &0x0205 ;r3 As==10
f9fa:   03 20           jnz $+8         ;abs 0xfa02
f9fc:   f2 d0 40 00     bis.b   #64,    &0x0021 ;#0x0040
fa00:   21 00
fa02:   92 c3 62 01     bic #1, &0x0162 ;r3 As==01
fa06:   00 13           reti
fa08:   b2 50 34 00     add #52,    &0x0172 ;#0x0034
fa0c:   72 01
fa0e:   b2 b0 00 10     bit #4096,  &0x0162 ;#0x1000
fa12:   62 01
fa14:   18 24           jz  $+50        ;abs 0xfa46
fa16:   c2 93 00 02     tst.b   &0x0200
fa1a:   0f 24           jz  $+32        ;abs 0xfa3a
```

```
fcf6:   0a 5e           add r14,    r10
fcf8:   0b 6f           addc    r15,    r11
fcfa:   0e 5e           rla r14
fcfc:   0f 6f           rlc r15
fcfe:   0d 93           tst r13
fd00:   f6 23           jnz $-18        ;abs 0xfcee
fd02:   0c 93           tst r12
fd04:   f4 23           jnz $-22        ;abs 0xfcee
fd06:   0c 4a           mov r10,    r12
fd08:   0d 4b           mov r11,    r13
fd0a:   3a 41           pop r10
fd0c:   30 41           ret
fd0e:   92 c3 64 01     bic #1, &0x0164 ;r3 As==01
fd12:   d2 93 01 02     cmp.b   #1, &0x0201 ;r3 As==01
fd16:   04 24           jz  $+10        ;abs 0xfd20
fd18:   f2 e0 41 00     xor.b   #65,    &0x0021 ;#0x0041
fd1c:   21 00
fd1e:   00 13           reti
fd20:   f2 f0 be 00     and.b   #190,   &0x0021 ;#0x00be
fd24:   21 00
fd26:   00 13           reti
fd28:   d2 c3 00 00     bic.b   #1, &0x0000 ;r3 As==01
fd2c:   d2 c3 02 00     bic.b   #1, &0x0002 ;r3 As==01
fd30:   b2 40 80 5a     mov #23168, &0x0120 ;#0x5a80
fd34:   20 01
fd36:   f2 d2 25 00     bis.b   #8, &0x0025 ;r2 As==11
fd3a:   00 13           reti
fd3c:   0f 4c           mov r12,    r15
fd3e:   0e 93           tst r14
fd40:   05 24           jz  $+12        ;abs 0xfd4c
```

```
[0x0000fcf8 98% 160 msp430code.bin]> pd $r @ fcn.0000f800+1272 # 0xfcf8
     ||   0x0000fcf8    0b6f           addc r15, r11
     ||   0x0000fcfa    0e5e           rla r14
     ||   0x0000fcfc    0f6f           rlc r15
     ||   0x0000fcfe    0d93           tst r13
     |`=< 0x0000fd00    f623           jnz $-0x0012                    ;[1]
     |    0x0000fd02    0c93           tst r12
     `==< 0x0000fd04    f423           jnz $-0x0016                    ;[1]
          0x0000fd06    0c4a           mov r10, r12
          0x0000fd08    0d4b           mov r11, r13
          0x0000fd0a    3a41           pop r10
          0x0000fd0c    3041           ret
          0x0000fd0e    92c36401       bic #1, &0x0164
          0x0000fd12    d2930102       cmp.b #1, &0x0201
    ,===< 0x0000fd16    0424           jeq $+0x000a                    ;[2]
    |     0x0000fd18    f2e041002100   xor.b #0x0041, &0x0021
    |     0x0000fd1e    0013           reti
    `---> 0x0000fd20    f2f0be002100   and.b #0x00be, &0x0021
          0x0000fd26    0013           reti
          0x0000fd28    d2c30000       bic.b #1, &0x0000
          0x0000fd2c    d2c30200       bic.b #1, &0x0002
          0x0000fd30    b240805a2001   mov #0x5a80, &0x0120
          0x0000fd36    f2d22500       eint
          0x0000fd3a    0013           reti
          0x0000fd3c    0f4c           mov r12, r15
          0x0000fd3e    0e93           tst r14
   ,====< 0x0000fd40    0524           jeq $+0x000c                    ;[3]
  .-----> 0x0000fd42    1f53           inc r15
  ||      0x0000fd44    ff4dffff1e83   mov.b @r13+, 0x831e(r15)
  `=====< 0x0000fd4a    fb23           jnz $-0x0008                    ;[4]
    `---> 0x0000fd4c    3041           ret
          0x0000fd4e    3441           pop r4
```

```
fcf8:   0b 6f           addc    r15,    r11
fcfa:   0e 5e           rla r14
fcfc:   0f 6f           rlc r15
fcfe:   0d 93           tst r13
fd00:   f6 23           jnz $-18        ;abs 0xfcee
fd02:   0c 93           tst r12
fd04:   f4 23           jnz $-22        ;abs 0xfcee
fd06:   0c 4a           mov r10,    r12
fd08:   0d 4b           mov r11,    r13
fd0a:   3a 41           pop r10
fd0c:   30 41           ret
fd0e:   92 c3 64 01     bic #1, &0x0164 ;r3 As==01
fd12:   d2 93 01 02     cmp.b   #1, &0x0201 ;r3 As==01
fd16:   04 24           jz  $+10        ;abs 0xfd20
fd18:   f2 e0 41 00     xor.b   #65,    &0x0021 ;#0x0041
fd1c:   21 00
fd1e:   00 13           reti
fd20:   f2 f0 be 00     and.b   #190,   &0x0021 ;#0x00be
fd24:   21 00
fd26:   00 13           reti
fd28:   d2 c3 00 00     bic.b   #1, &0x0000 ;r3 As==01
fd2c:   d2 c3 02 00     bic.b   #1, &0x0002 ;r3 As==01
fd30:   b2 40 80 5a     mov #23168, &0x0120 ;#0x5a80
fd34:   20 01
fd36:   f2 d2 25 00     bis.b   #8, &0x0025 ;r2 As==11
fd3a:   00 13           reti
fd3c:   0f 4c           mov r12,    r15
fd3e:   0e 93           tst r14
fd40:   05 24           jz  $+12        ;abs 0xfd4c
```

Radio Modules

- IOT, “Smart” everything
- Communicate with MCU through standard communication busses

Obvious Methods

- Frequency printed on device

Obvious Methods

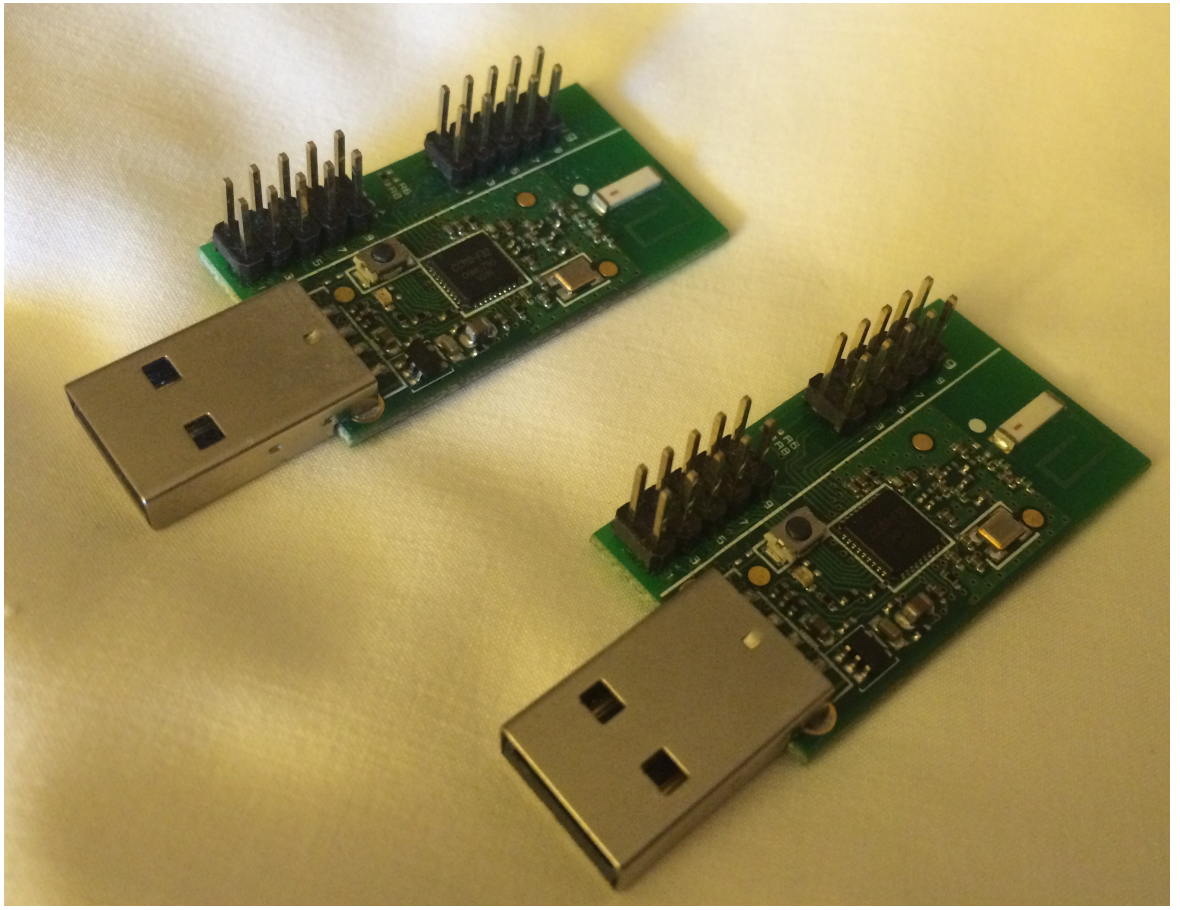
- Frequency printed on device
- Frequency printed in instruction manuals

Obvious Methods

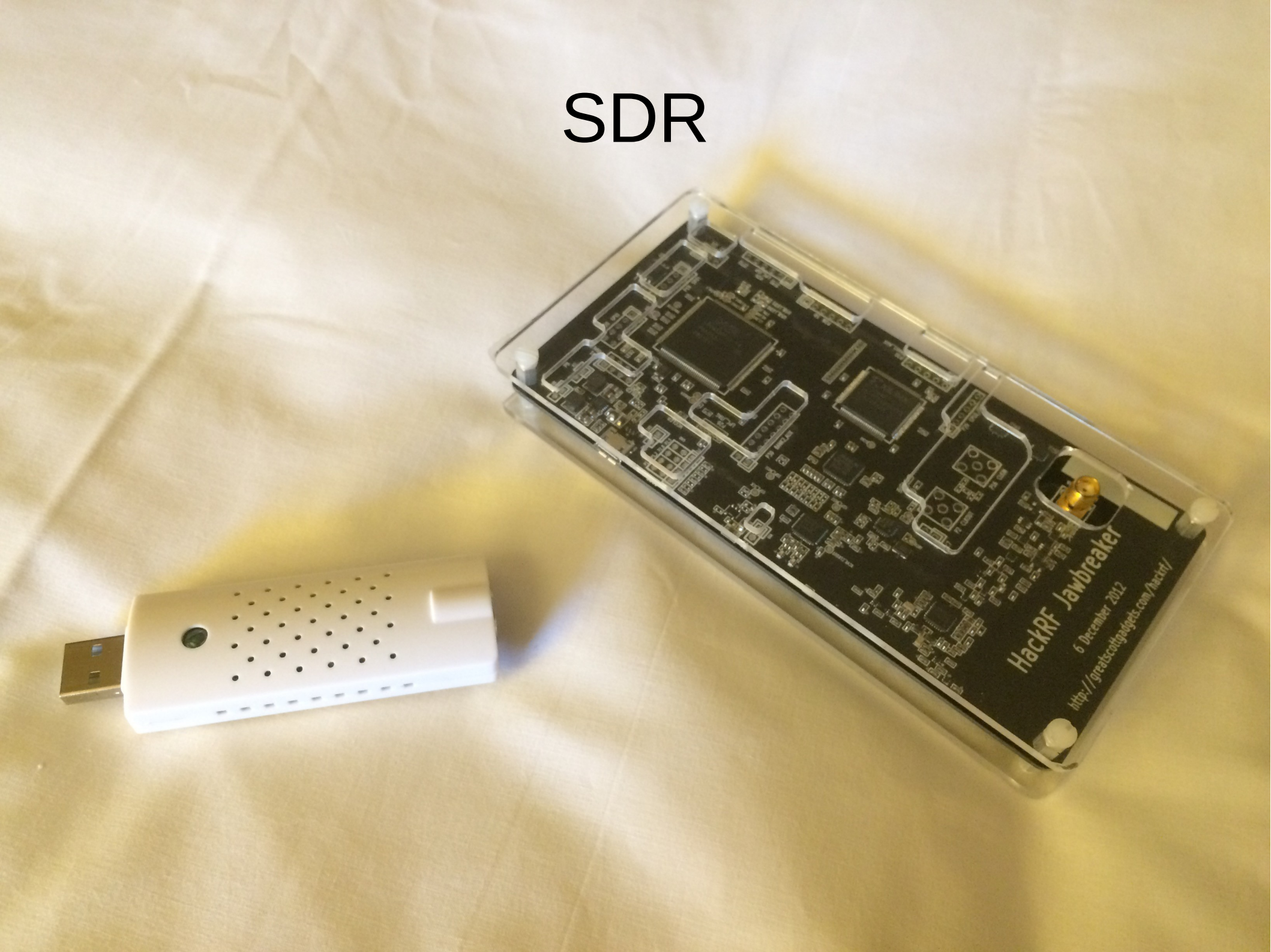
- Frequency printed on device
- Frequency printed in instruction manuals
- FCC documentation

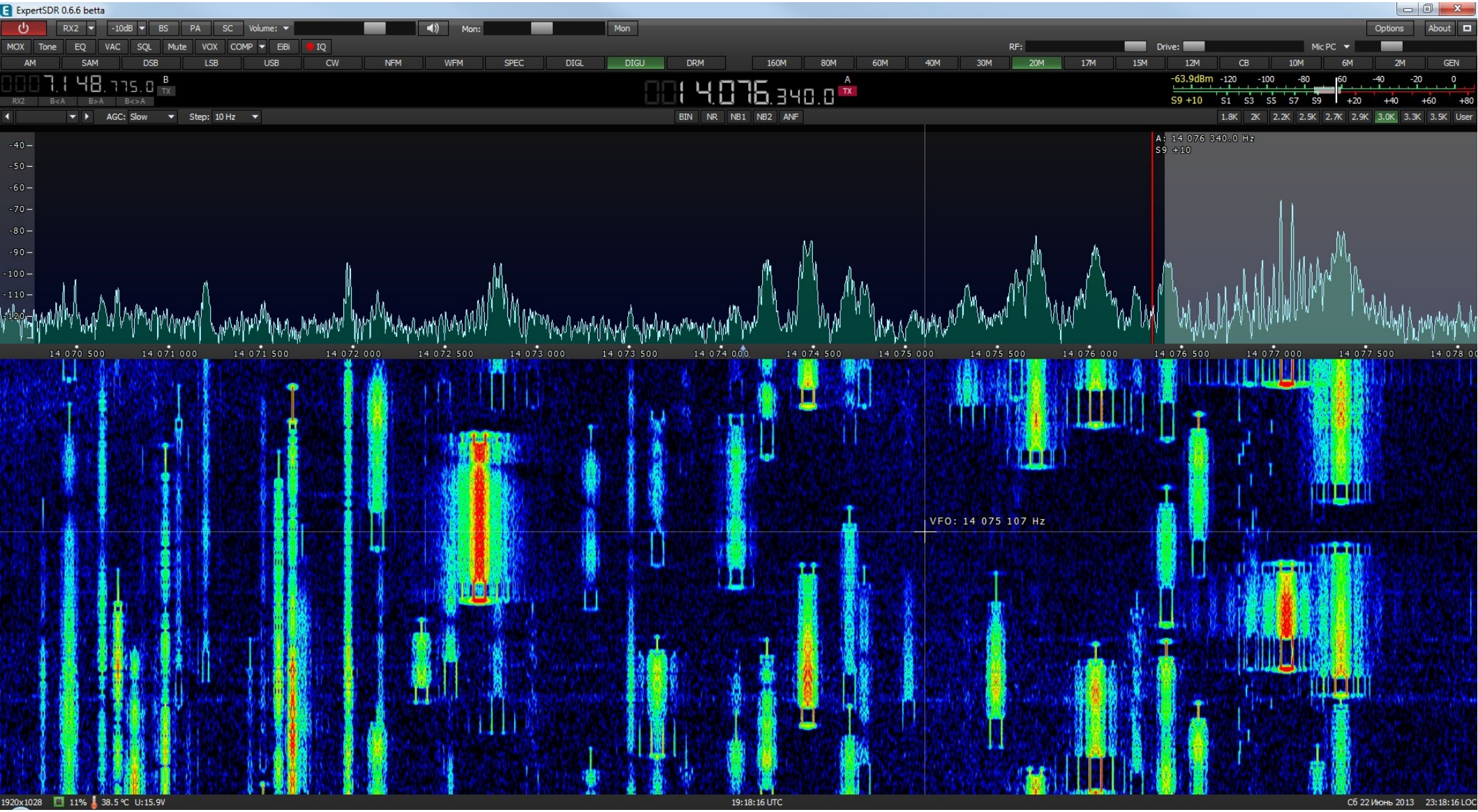
RFCat

- Rx / Tx of custom packets
- Manually set frequency, modulation, etc.

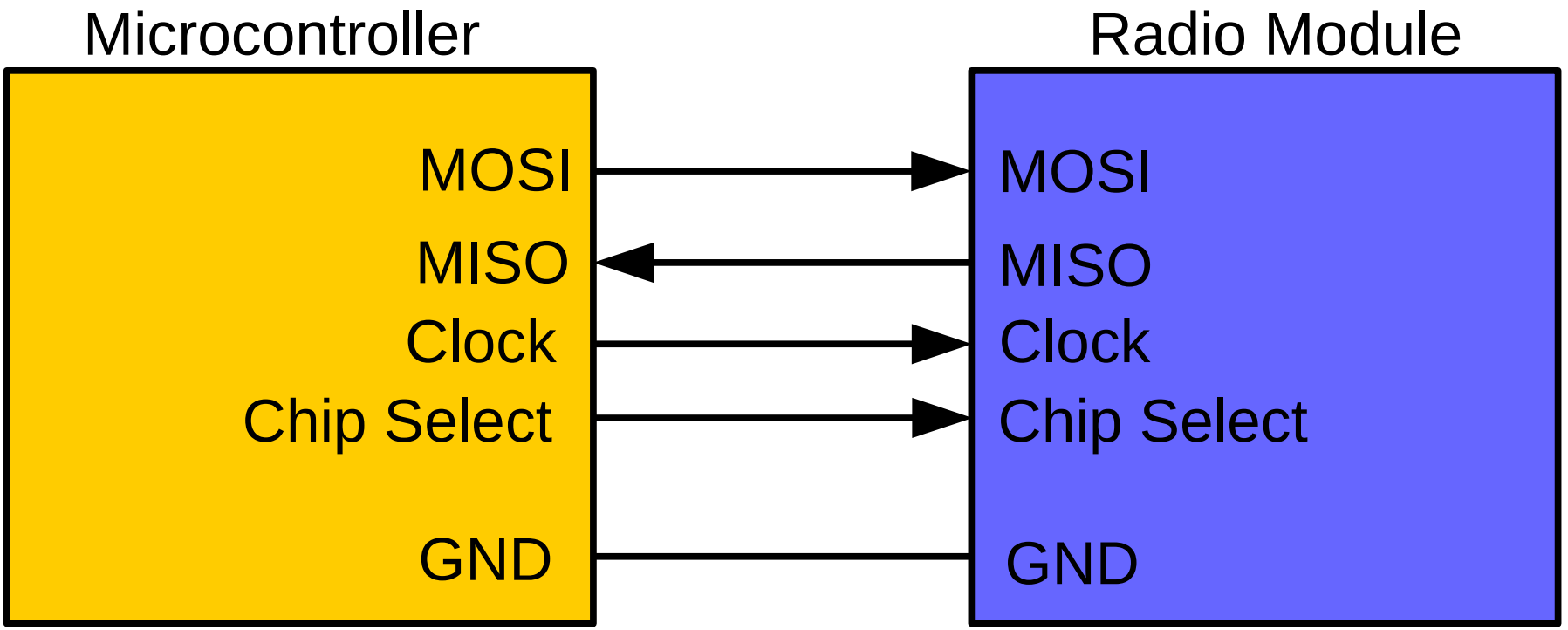


SDR


HackRF Jawbreaker
6 December 2012
http://greatscottgadgets.com/hackrf/


ExpertSDR 0.6.6 betta
RX2
-10dB
BS
PA
SC
Volume:
Mon:
Mon
Options
About
MOX
Tone
EQ
VAC
SQL
Mute
VOX
COMP
EiBi
IQ
RF:
Drive:
Mic PC
AM
SAM
DSB
LSB
USB
CW
NFM
WFM
SPEC
DIGL
DIGU
DRM
160M
80M
60M
40M
30M
20M
17M
15M
12M
CB
10M
6M
2M
GEN
14.076.340.0
-63.9dBm
S9 +10
AGC: Slow
Step: 10 Hz
BIN
NR
NB1
NB2
ANF
A: 14 076 340.0 Hz
S9 +10
VFO: 14 075 107 Hz
14 070 500
14 071 000
14 071 500
14 072 000
14 072 500
14 073 000
14 073 500
14 074 000
14 074 500
14 075 000
14 075 500
14 076 000
14 076 500
14 077 000
14 077 500
14 078 00
1920x1028
11%
38.5 °C
U:15.9V
19:18:16 UTC
Сб 22 Июнь 2013
23:18:16 LOC

Bus Sniffing

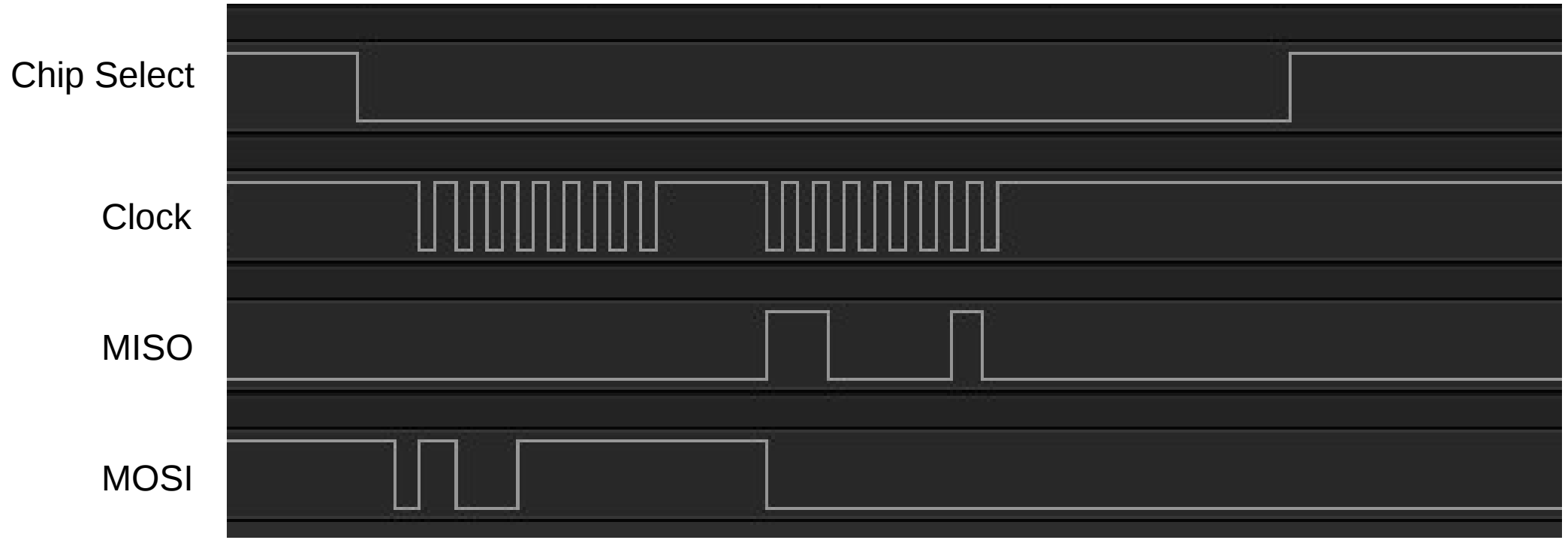


Bus Sniffing

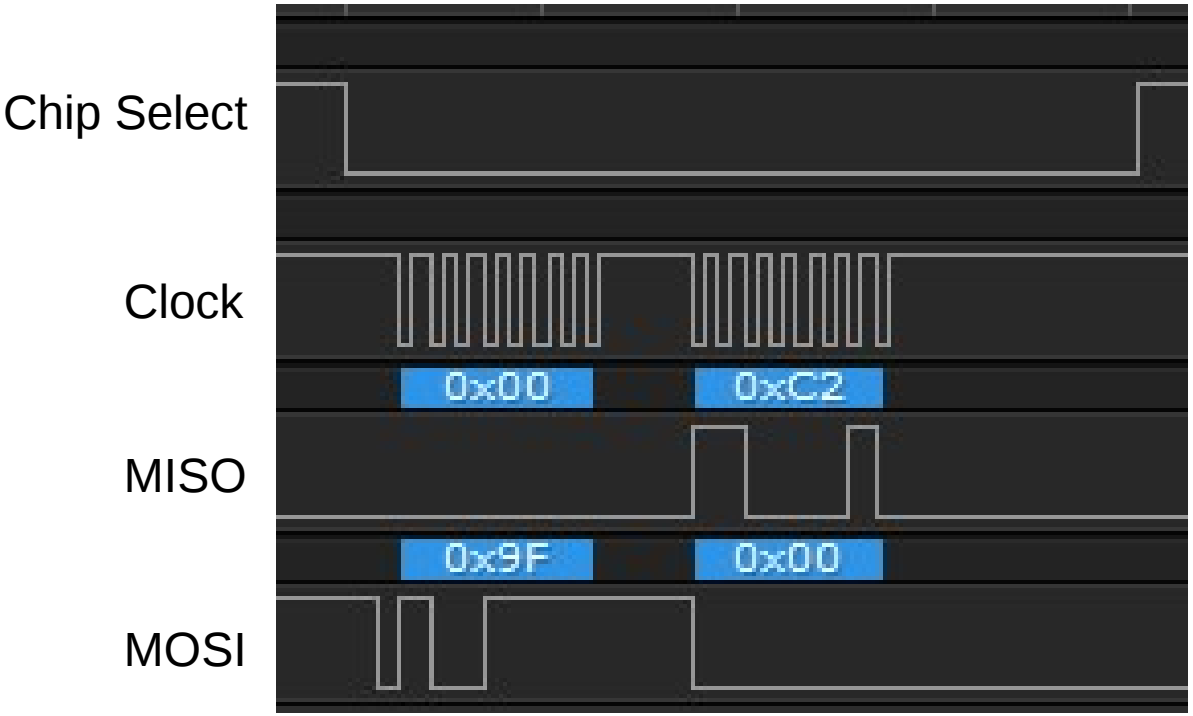
	868 MHz		915 MHz	
Output Power [dBm]	**Setting**	**Current Consumption, Typ. [mA]**	**Setting**	**Current Consumption, Typ. [mA]**
-30	0x03	12.0	0x03	11.9
-20	0x17	12.6	0x0E	12.5
-15	0x1D	13.3	0x1E	13.3
-10	0x26	14.5	0x27	14.8
-6	0x37	16.4	0x38	17.0
0	0x50	16.8	0x8E	17.2
5	0x86	19.9	0x84	20.2
7	0xCD	25.8	0xCC	25.7
10	0xC5	30.0	0xC3	30.7
12/11	0xC0	34.2	0xC0	33.4

Bus Sniffing

Bus Sniffing



Bus Sniffing

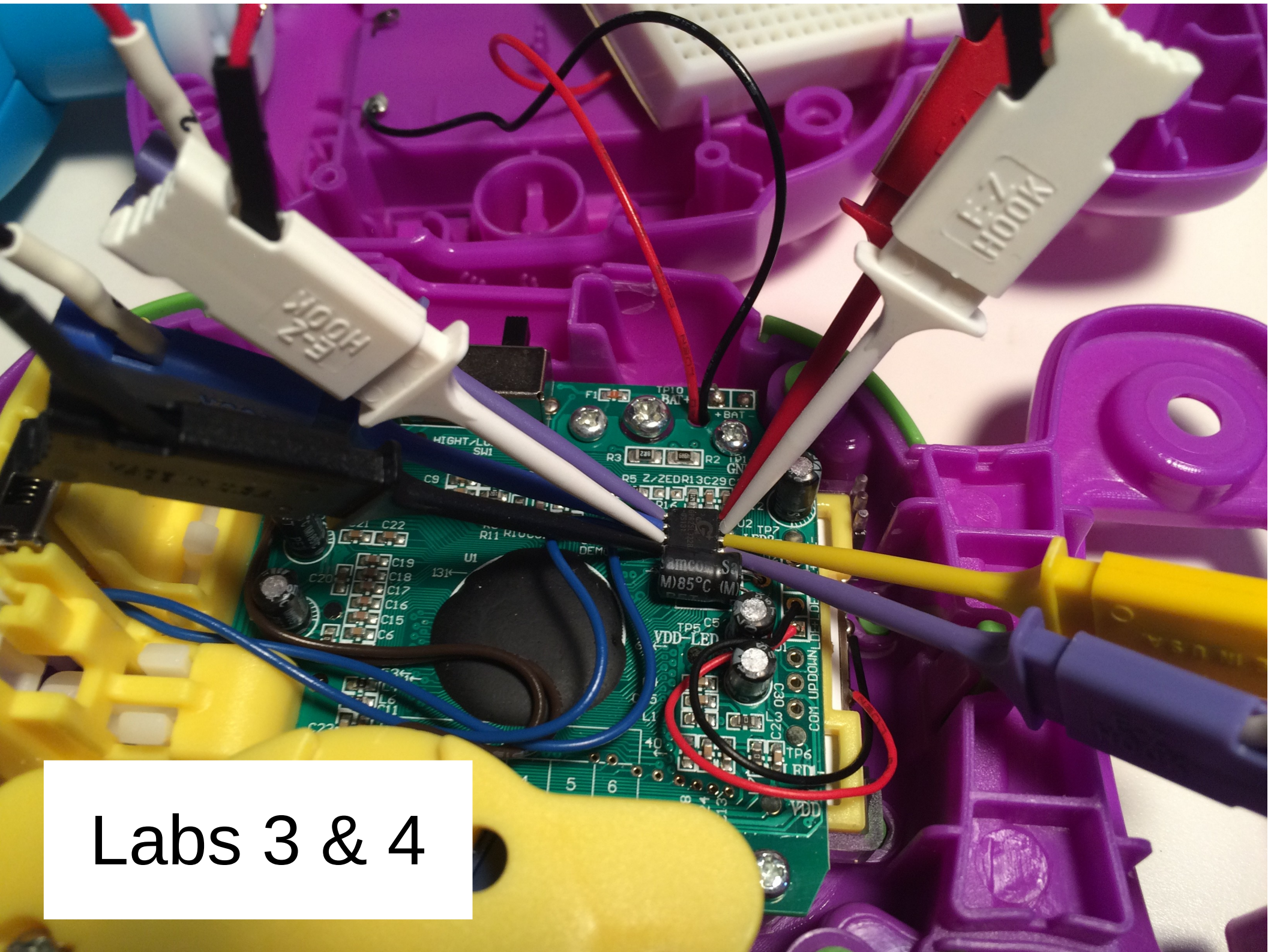


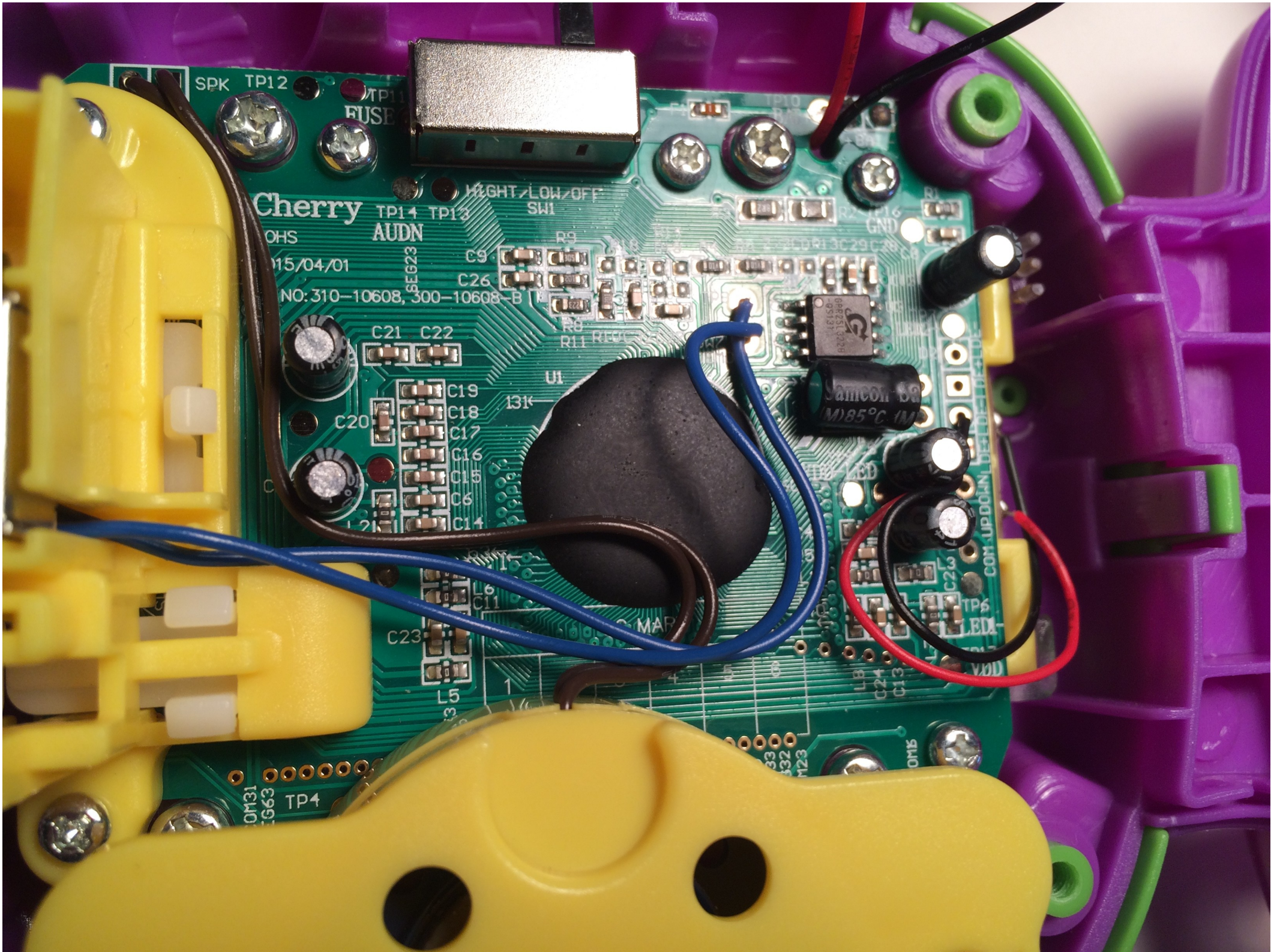
Logic Analyzers

Intronix Logicport
34 channels, 200-500 MSPS
Amazing trigger options, ~$400

QuantAsylum QA101
12-24 channels, 30 MSPS
Oscilloscope, wave gen, ~$400

Saleae Logic
4-16 channels, 12-500 MSPS
$110 - $600


Labs 3 & 4

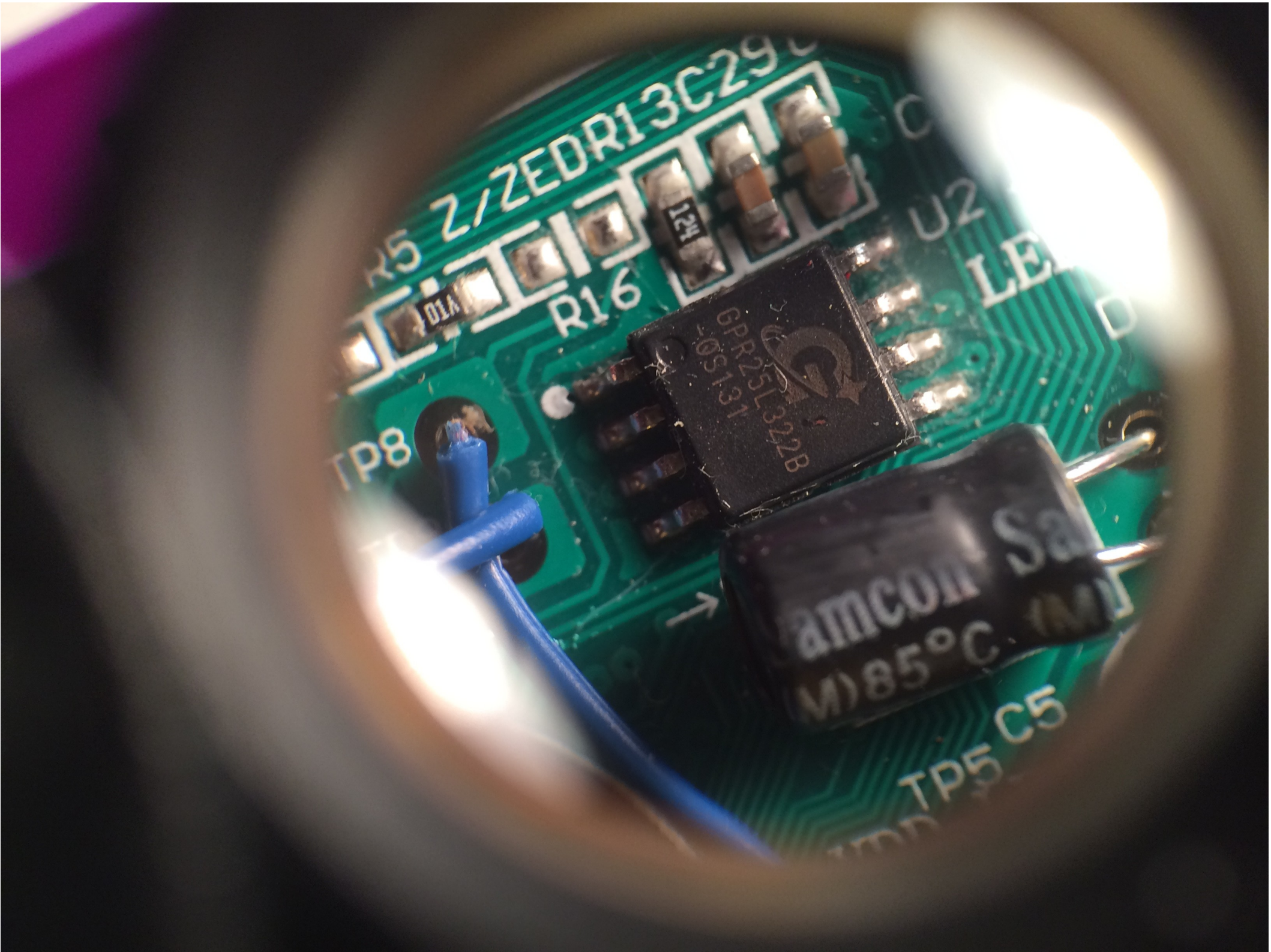

SPK TP12
Cherry
TP14 TP13
AUDN
HIGHT/LOW/OFF
SW1
NO:310-10608, 300-10608-B
C21 C22
C19
C18
C17
C16
C20
U1
TP4
TP6
LED1
VDD

- Red dot on chip

- Quality Control?
- Prevent reverse engineering?

- Red dot on chip


MG Chemicals
HEAVY DUTY
FLUX
REMOVER
• Aggressively dissolves all types of flux, including burnt on flux
• Harmful to some plastics
• Dissout agressivement tous les types de flux, incluant le flux brûlé
• Nocive pour certains plastiques
• Agresivamente disuelve todos los tip de flujo, incluyendo el flujo quemad
• Perjudiciales para algunos plásticos
UX REMOVER
O. 413B-425G
425 g
DANGER. PELIGRO.
MMABLE. INFLAMMABLE. INFLAMABLE. POISON.
ENO. CONTENTS UNDER PRESSURE. CONTENU
OUS PRESSION. CONTENIDO BAJO PRESIÓN.
cautions on back panel. Lisez les attentions sur le panneau
arrière. Leer advertencias de tras de la etiqueta.


R16
U2
TP8
TP5
C5
GPR25L322B
0S131
M)85°C

DATA SHEET

GPR25L322B

32M-BIT [x1 / x2] CMOS SERIAL FLASH

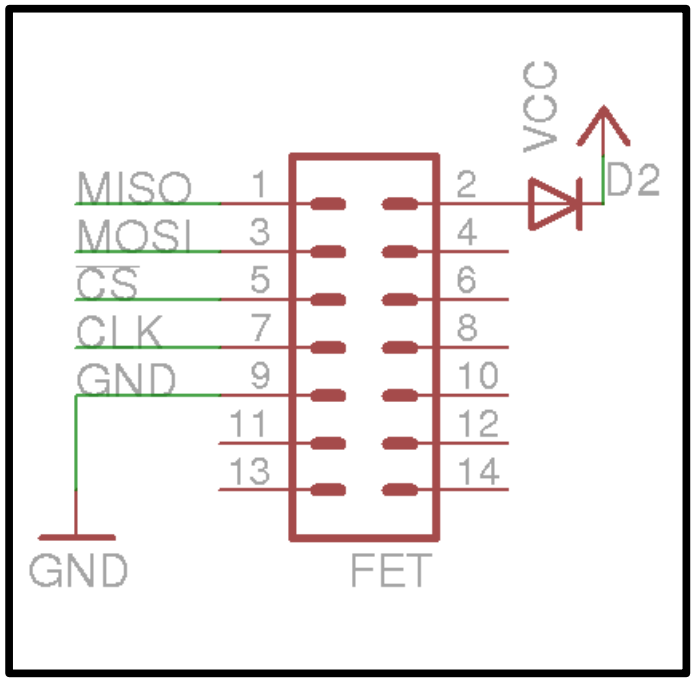
Jun. 06, 2014

Version 1.3

Lab 3

Using GoodFET pinout & SPI flash datasheet, extract data from Word Whammer flash module

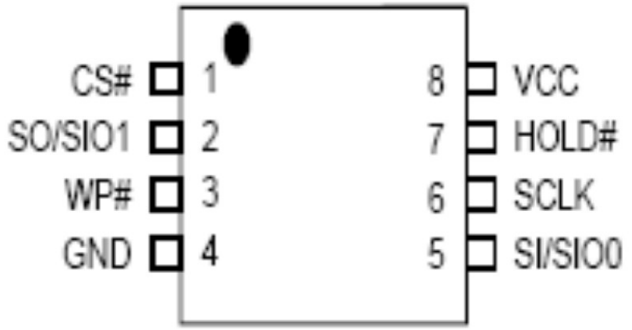
```
# sha1sum dumpWordWhammerFlash.bin

1758376c6673555fe63acff3c1ebcc715237l2fb
```



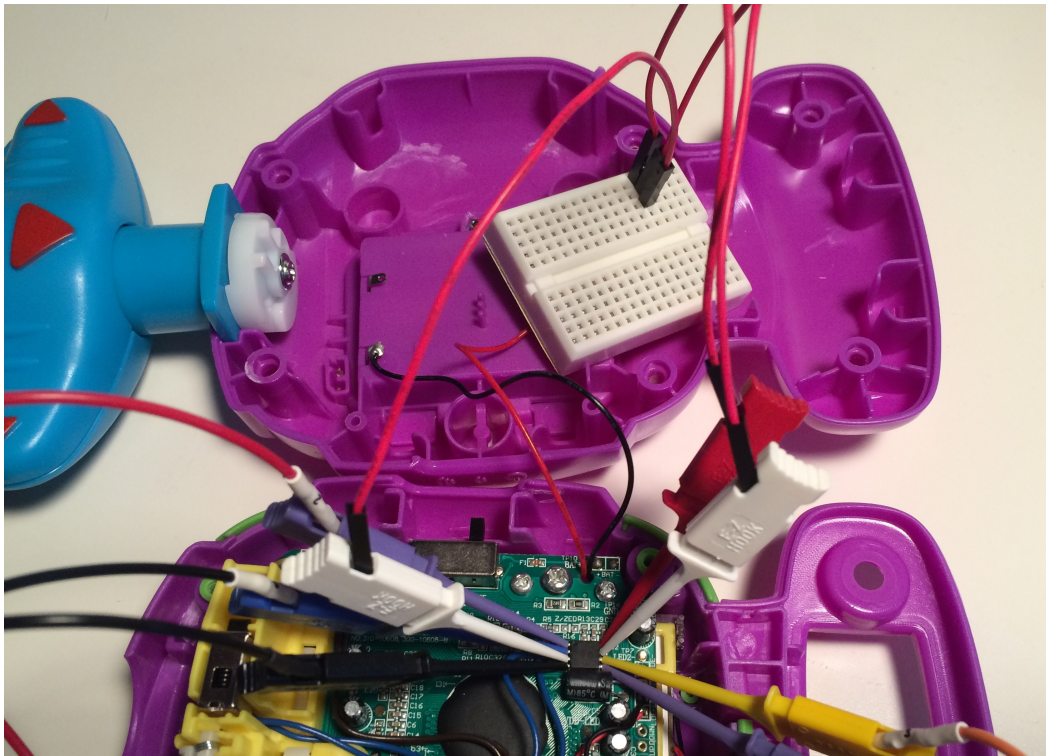
GPR25L322B

3. PIN CONFIGURATIONS

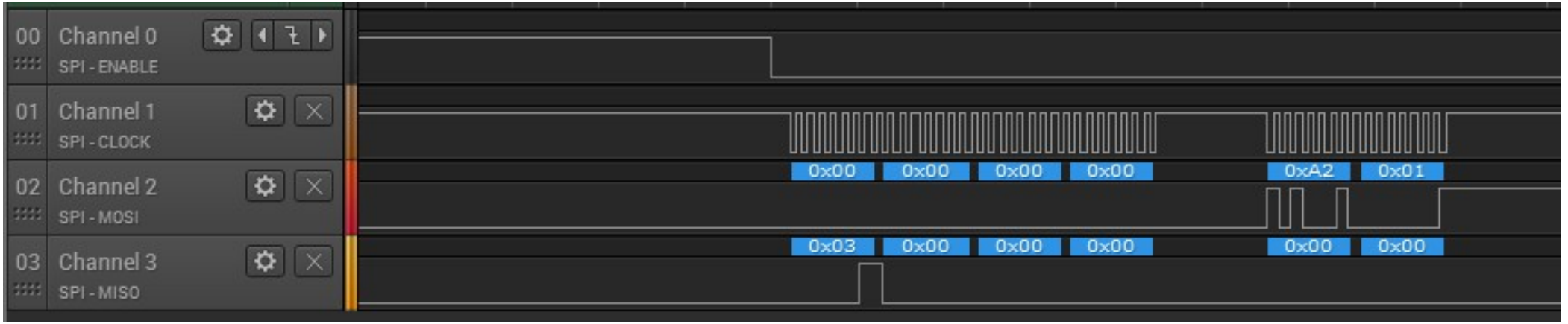
3.1. 8-PIN SOP (209mil)



Use breadboard to pull WP# and HOLD# to VCC

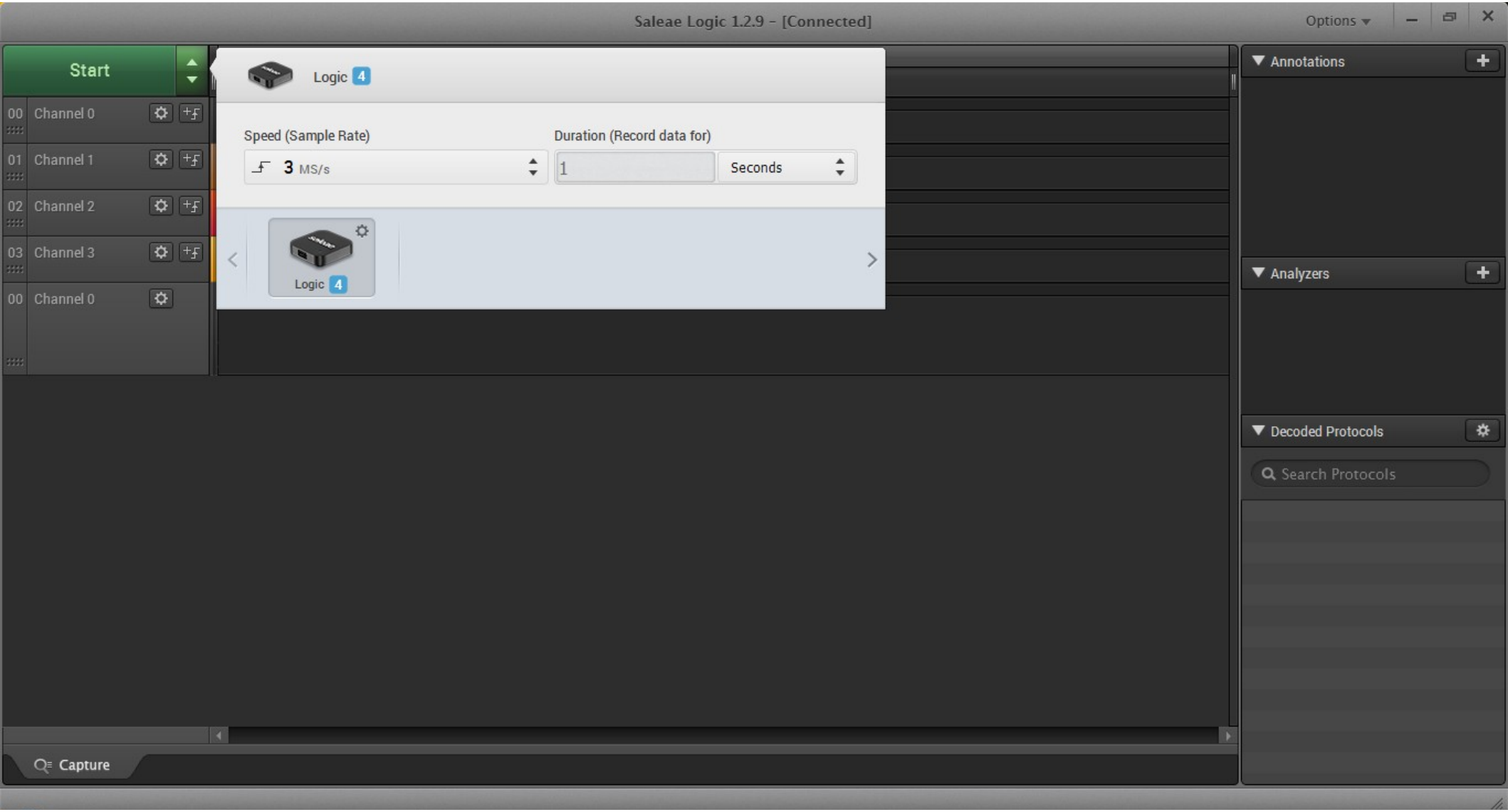


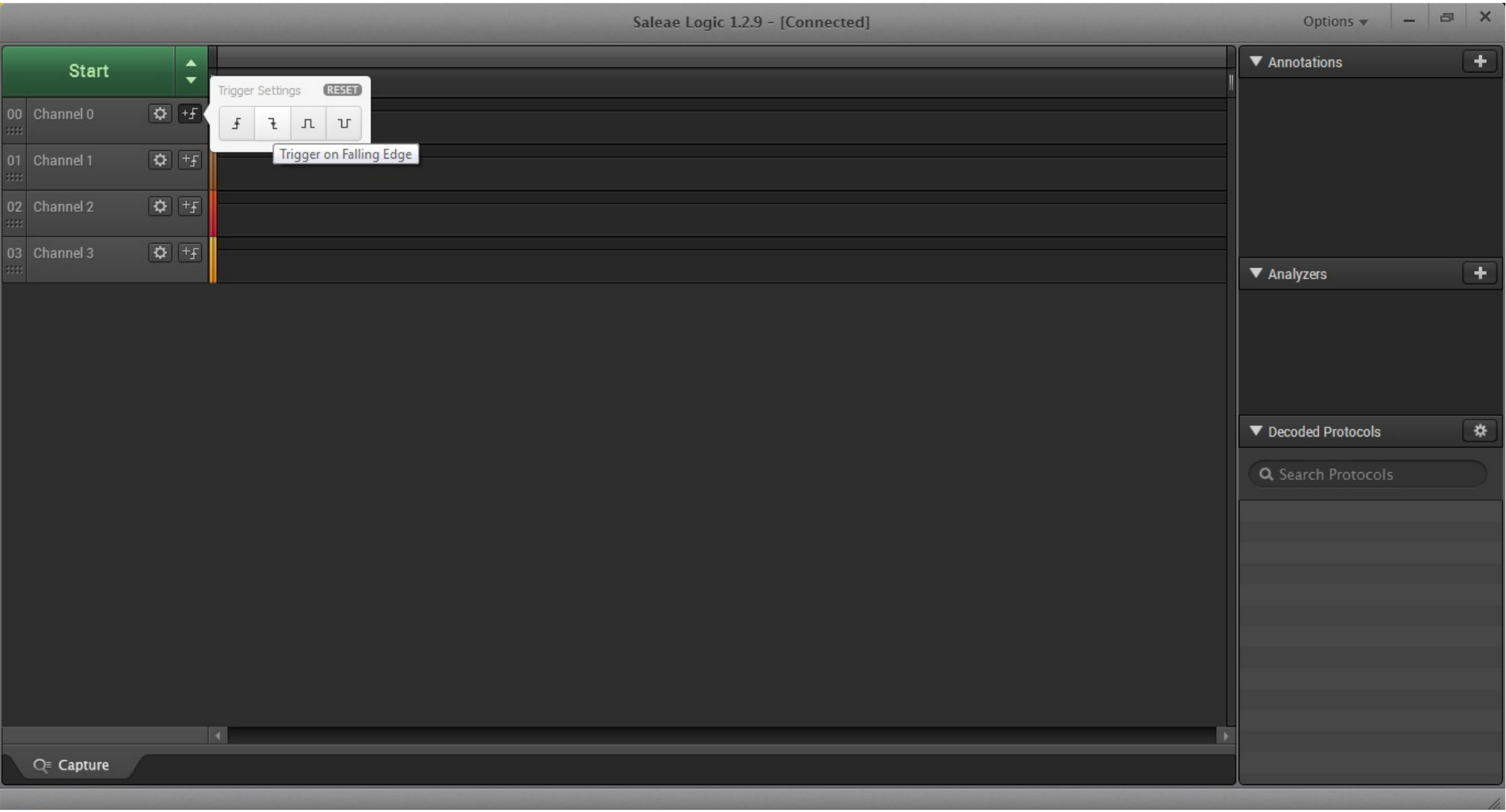
Lab 4

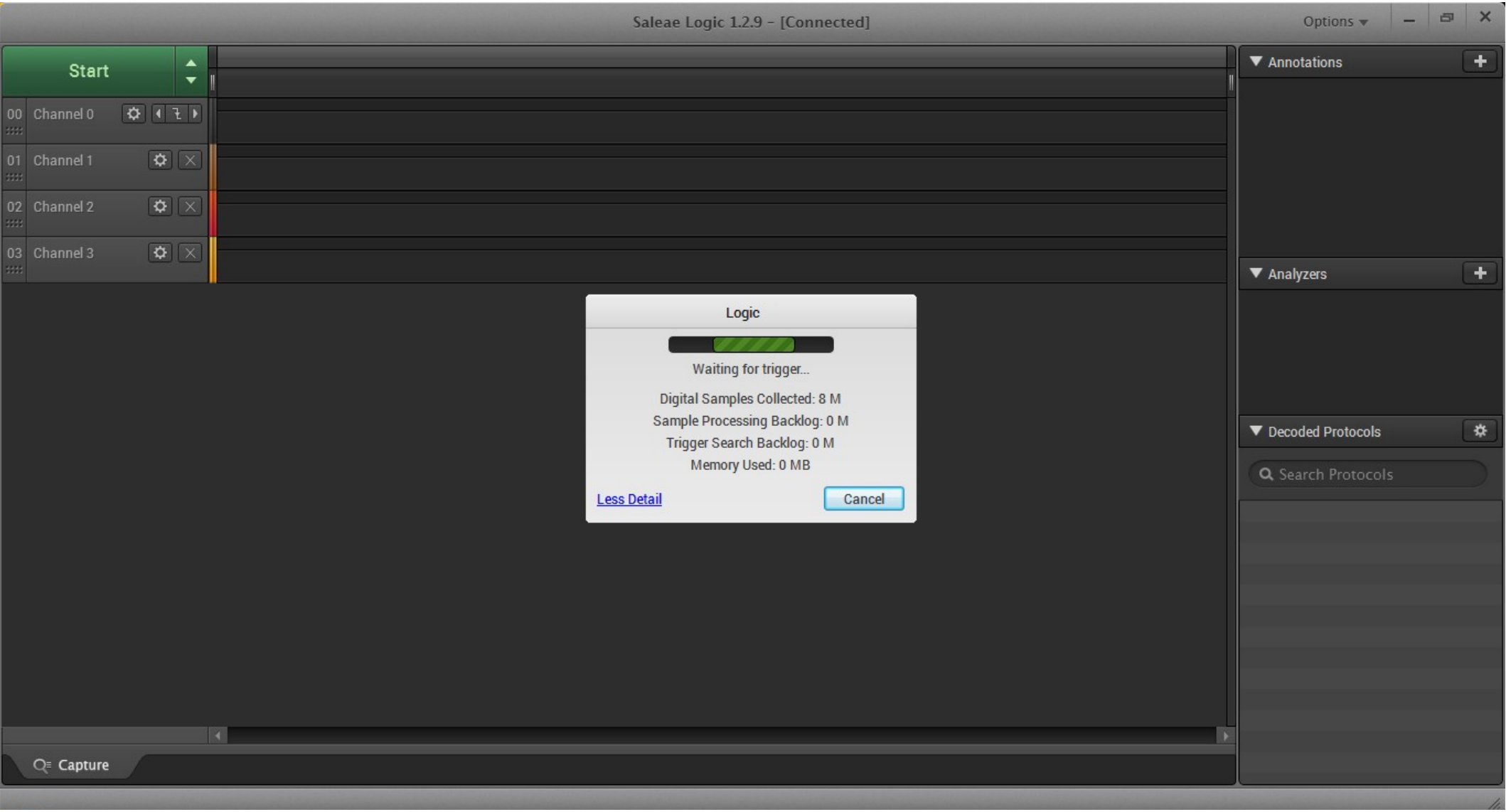

00 Channel 0
SPI - ENABLE
01 Channel 1
SPI - CLOCK
02 Channel 2
SPI - MOSI
03 Channel 3
SPI - MISO
0x00
0x00
0x00
0x00
0xA2
0x01
0x03
0x00
0x00
0x00
0x00
0x00

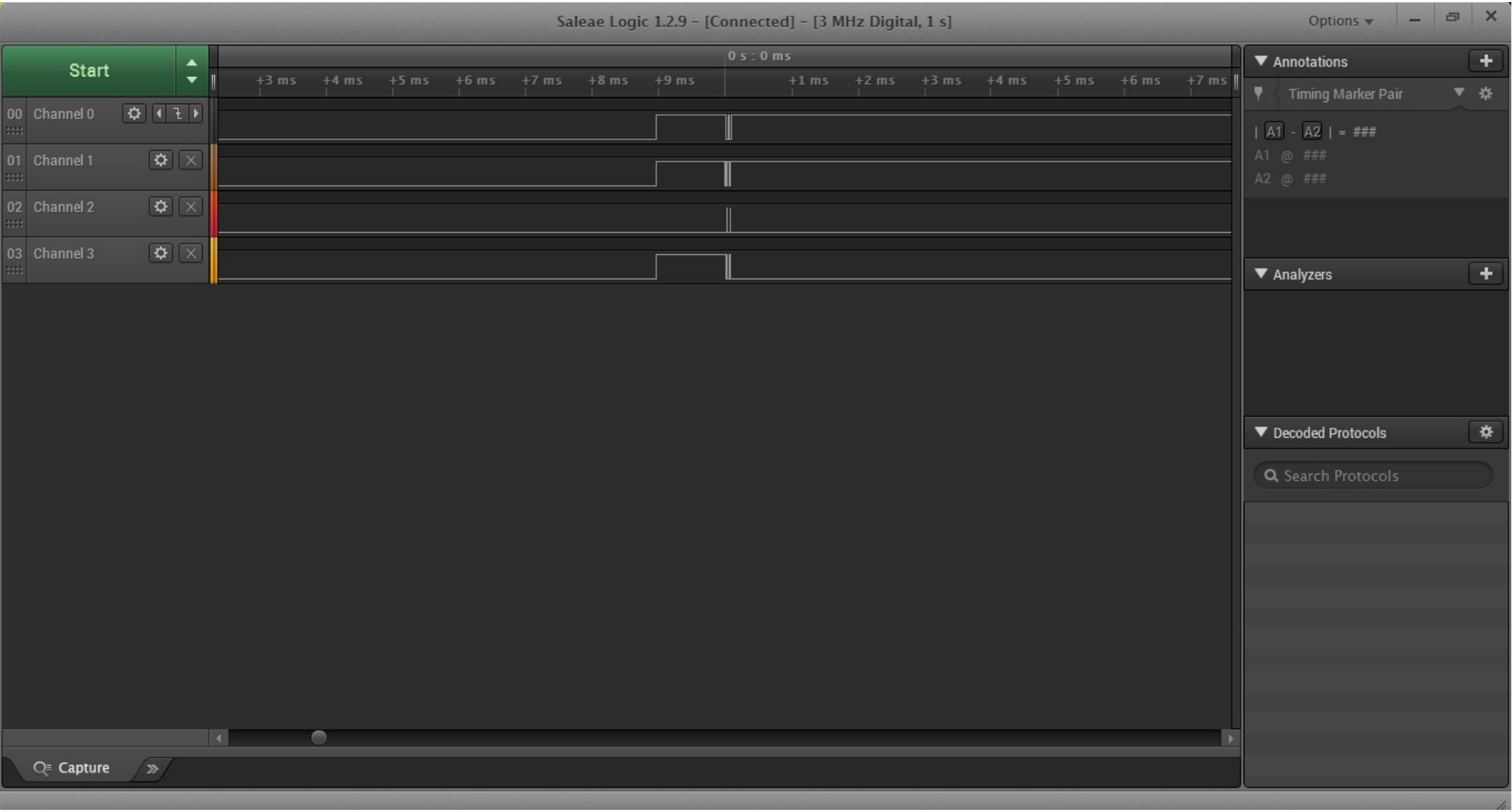
Saleae Logic Software

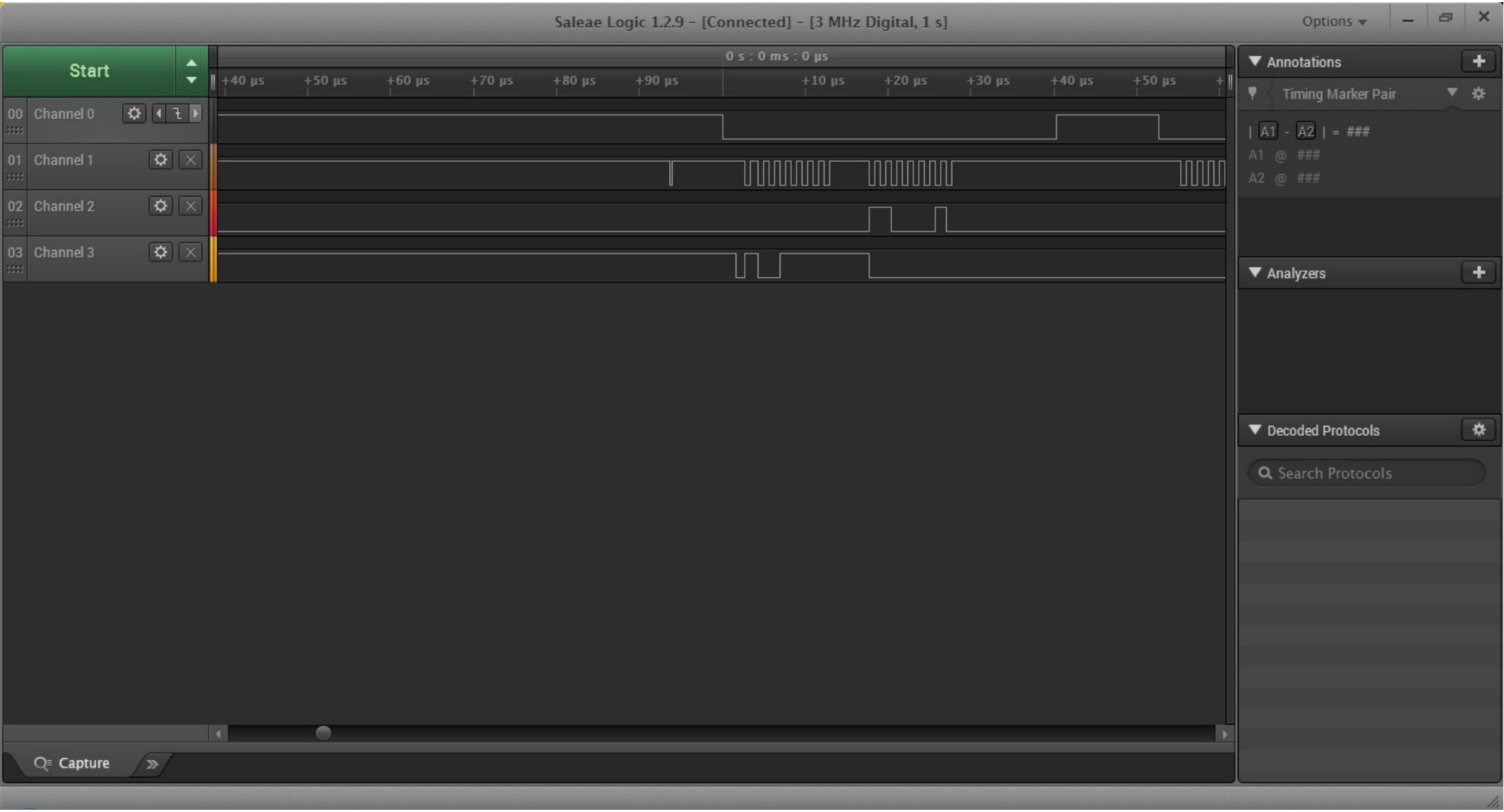
- https://www.saleae.com/downloads
- Runs natively in Windows, OS X, and Linux
  - Linux users, run as root

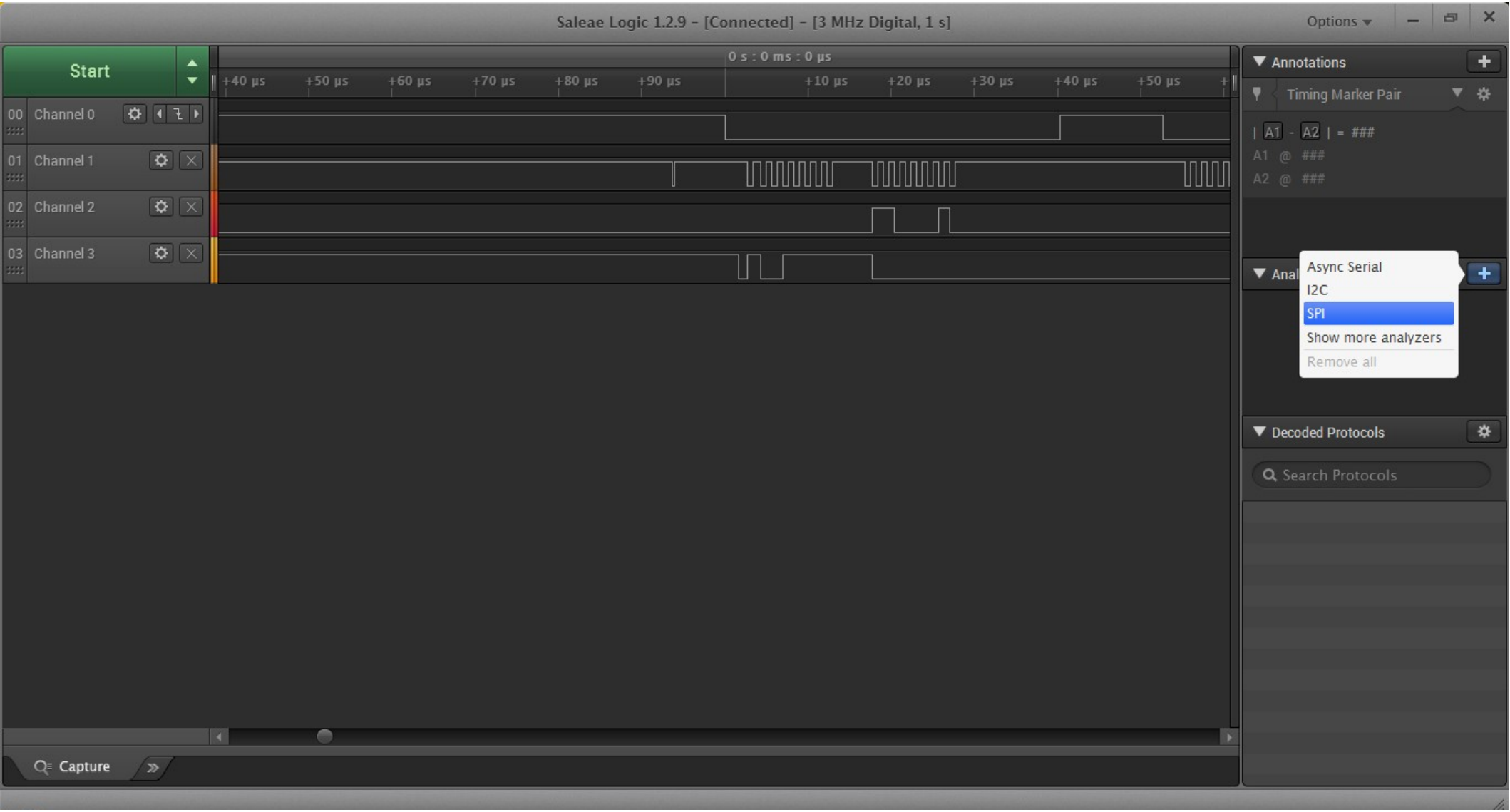

Saleae Logic 1.2.9 - [Connected]
Options
Start
Logic 4
00 Channel 0
01 Channel 1
02 Channel 2
03 Channel 3
00 Channel 0
Speed (Sample Rate)
3 MS/s
Duration (Record data for)
1
Seconds
Logic 4
Annotations
Analyzers
Decoded Protocols
Search Protocols
Capture

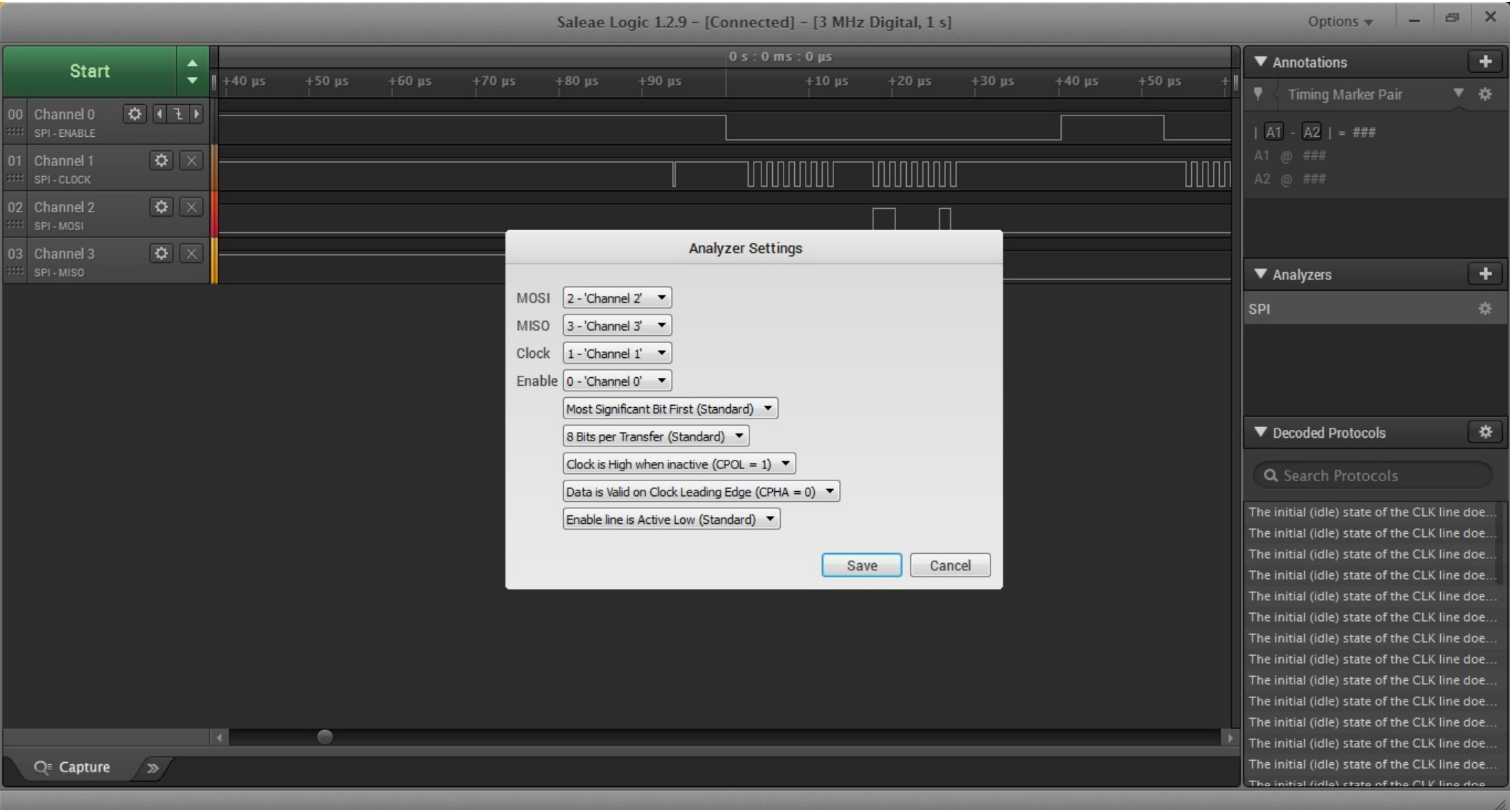

Saleae Logic 1.2.9 - [Connected]
Options
Start
Trigger Settings
RESET
00 Channel 0
01 Channel 1
02 Channel 2
03 Channel 3
Trigger on Falling Edge
Annotations
Analyzers
Decoded Protocols
Search Protocols
Capture

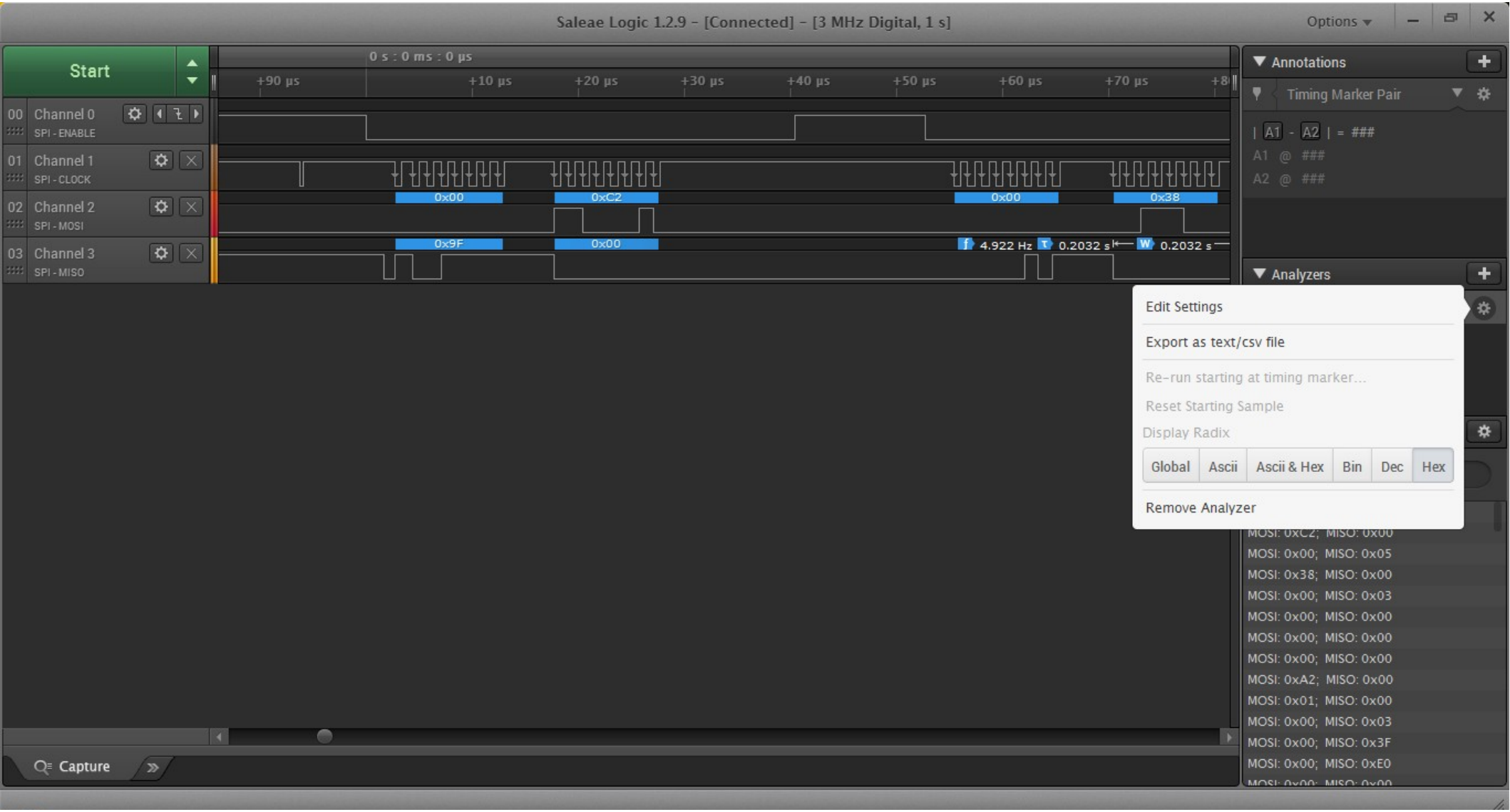

Saleae Logic 1.2.9 - [Connected]
Options
Start
00 Channel 0
01 Channel 1
02 Channel 2
03 Channel 3
Annotations
Analyzers
Decoded Protocols
Search Protocols
Logic
Waiting for trigger...
Digital Samples Collected: 8 M
Sample Processing Backlog: 0 M
Trigger Search Backlog: 0 M
Memory Used: 0 MB
Less Detail
Cancel
Capture

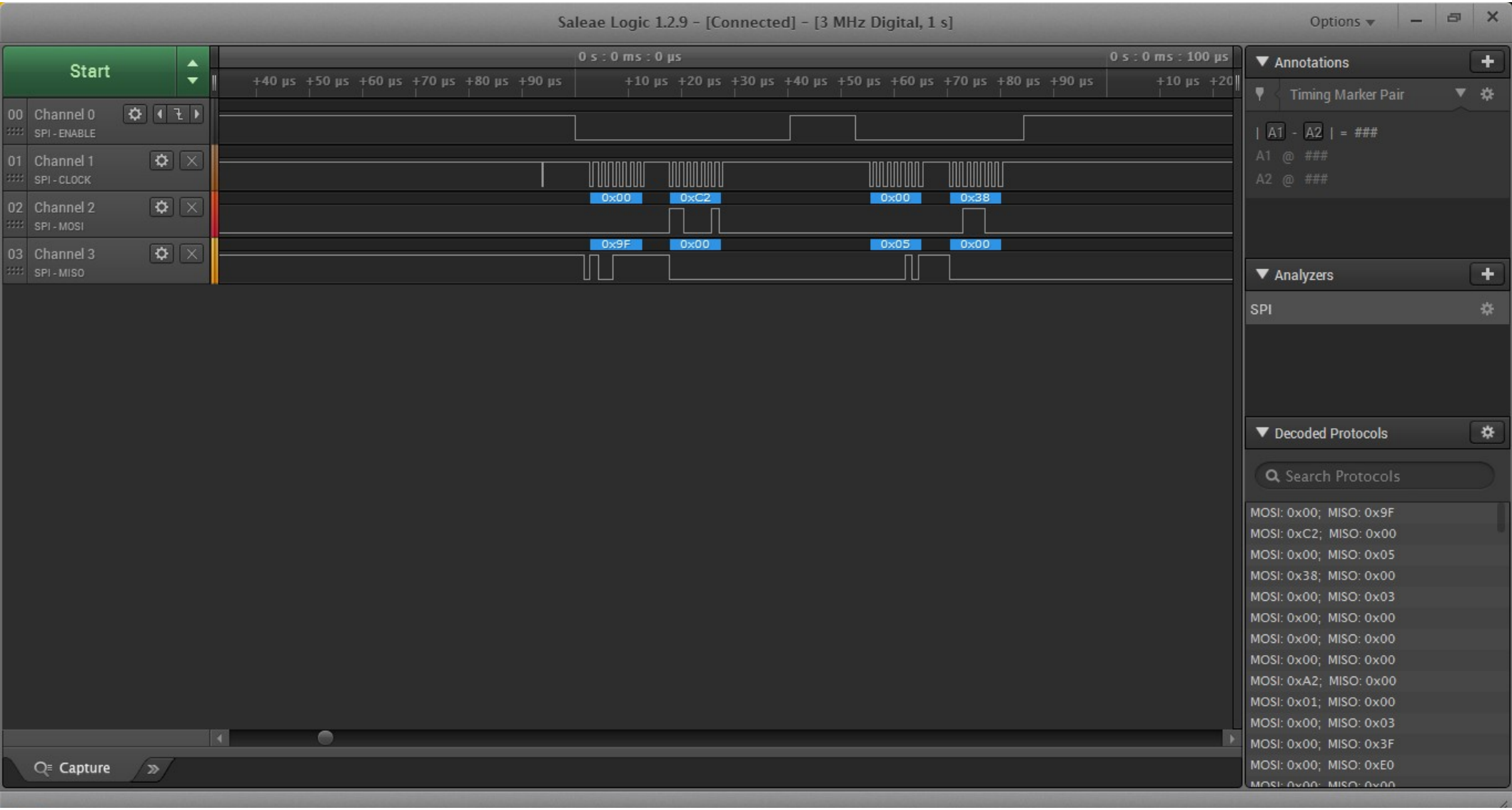

Saleae Logic 1.2.9 - [Connected] - [3 MHz Digital, 1 s]
Options
Start
0 s : 0 ms
+3 ms
+4 ms
+5 ms
+6 ms
+7 ms
+8 ms
+9 ms
+1 ms
+2 ms
+3 ms
+4 ms
+5 ms
+6 ms
+7 ms
00 Channel 0
01 Channel 1
02 Channel 2
03 Channel 3
Annotations
Timing Marker Pair
| A1 - A2 | = ###
A1 @ ###
A2 @ ###
Analyzers
Decoded Protocols
Search Protocols
Capture

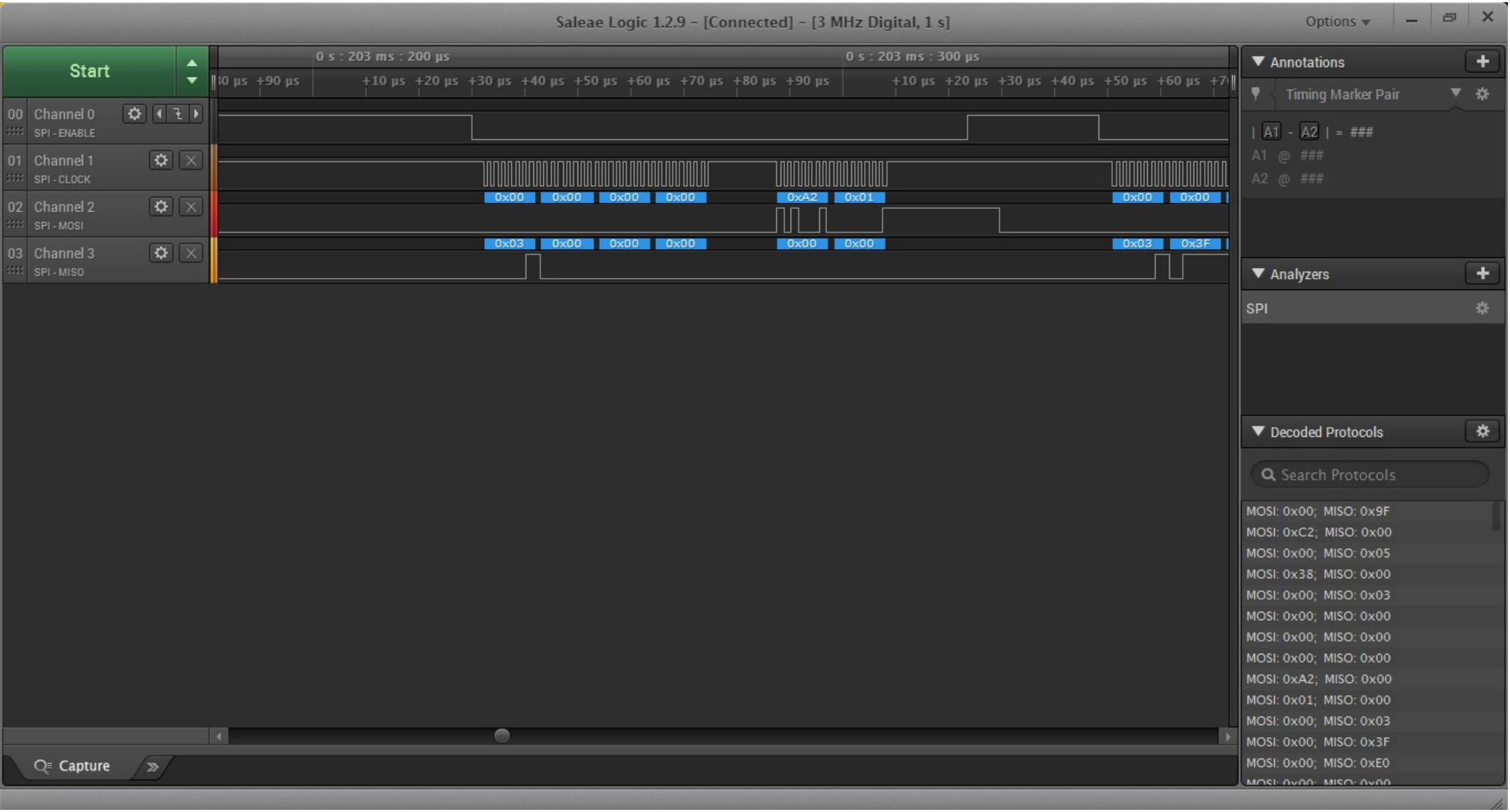

Saleae Logic 1.2.9 - [Connected] - [3 MHz Digital, 1 s]
Options
Start
0 s : 0 ms : 0 µs
+40 µs
+50 µs
+60 µs
+70 µs
+80 µs
+90 µs
+10 µs
+20 µs
+30 µs
+40 µs
+50 µs
00 Channel 0
01 Channel 1
02 Channel 2
03 Channel 3
Annotations
Timing Marker Pair
| A1 - A2 | = ###
A1 @ ###
A2 @ ###
Analyzers
Decoded Protocols
Search Protocols
Capture

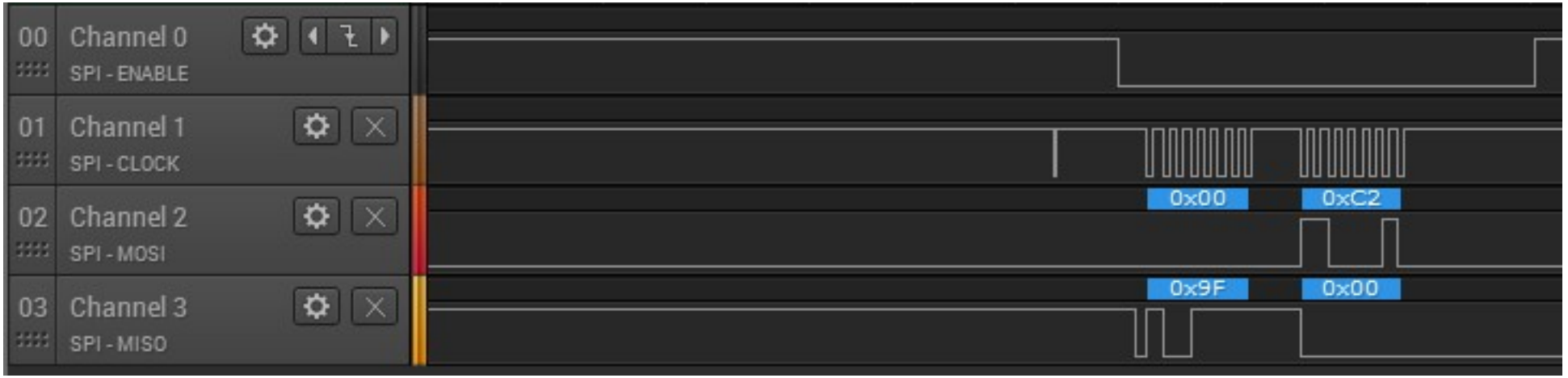

Saleae Logic 1.2.9 - [Connected] - [3 MHz Digital, 1 s]
Options
Start
0 s : 0 ms : 0 µs
+40 µs
+50 µs
+60 µs
+70 µs
+80 µs
+90 µs
+10 µs
+20 µs
+30 µs
+40 µs
+50 µs
00 Channel 0
01 Channel 1
02 Channel 2
03 Channel 3
Annotations
Timing Marker Pair
| A1 - A2 | = ###
A1 @ ###
A2 @ ###
Async Serial
I2C
SPI
Show more analyzers
Remove all
Decoded Protocols
Search Protocols
Capture


Saleae Logic 1.2.9 - [Connected] - [3 MHz Digital, 1 s]
Options
Start
0 s : 0 ms : 0 µs
+40 µs
+50 µs
+60 µs
+70 µs
+80 µs
+90 µs
+10 µs
+20 µs
+30 µs
+40 µs
+50 µs
00 Channel 0
SPI - ENABLE
01 Channel 1
SPI - CLOCK
02 Channel 2
SPI - MOSI
03 Channel 3
SPI - MISO
Analyzer Settings
MOSI
2 - 'Channel 2'
MISO
3 - 'Channel 3'
Clock
1 - 'Channel 1'
Enable
0 - 'Channel 0'
Most Significant Bit First (Standard)
8 Bits per Transfer (Standard)
Clock is High when inactive (CPOL = 1)
Data is Valid on Clock Leading Edge (CPHA = 0)
Enable line is Active Low (Standard)
Save
Cancel
Annotations
Timing Marker Pair
| A1 - A2 | = ###
A1 @ ###
A2 @ ###
Analyzers
SPI
Decoded Protocols
Search Protocols
The initial (idle) state of the CLK line doe...
Capture


Saleae Logic 1.2.9 - [Connected] - [3 MHz Digital, 1 s]
Options
Start
0 s : 0 ms : 0 µs
+90 µs
+10 µs
+20 µs
+30 µs
+40 µs
+50 µs
+60 µs
+70 µs
00 Channel 0
SPI - ENABLE
01 Channel 1
SPI - CLOCK
02 Channel 2
SPI - MOSI
03 Channel 3
SPI - MISO
0x00
0xC2
0x00
0x38
0x9F
0x00
4.922 Hz
0.2032 s
0.2032 s
Annotations
Timing Marker Pair
| A1 - A2 | = ###
A1 @ ###
A2 @ ###
Analyzers
Edit Settings
Export as text/csv file
Re-run starting at timing marker...
Reset Starting Sample
Display Radix
Global
Ascii
Ascii & Hex
Bin
Dec
Hex
Remove Analyzer
MOSI: 0xC2; MISO: 0x00
MOSI: 0x00; MISO: 0x05
MOSI: 0x38; MISO: 0x00
MOSI: 0x00; MISO: 0x03
MOSI: 0x00; MISO: 0x00
MOSI: 0x00; MISO: 0x00
MOSI: 0x00; MISO: 0x00
MOSI: 0xA2; MISO: 0x00
MOSI: 0x01; MISO: 0x00
MOSI: 0x00; MISO: 0x03
MOSI: 0x00; MISO: 0x3F
MOSI: 0x00; MISO: 0xE0
Capture


Saleae Logic 1.2.9 - [Connected] - [3 MHz Digital, 1 s]
Options
Start
0 s : 0 ms : 0 µs
0 s : 0 ms : 100 µs
+40 µs +50 µs +60 µs +70 µs +80 µs +90 µs
+10 µs +20 µs +30 µs +40 µs +50 µs +60 µs +70 µs +80 µs +90 µs
+10 µs +20
00 Channel 0
SPI - ENABLE
01 Channel 1
SPI - CLOCK
02 Channel 2
SPI - MOSI
03 Channel 3
SPI - MISO
0x00
0xC2
0x00
0x38
0x9F
0x00
0x05
0x00
Annotations
Timing Marker Pair
| A1 - A2 | = ###
A1 @ ###
A2 @ ###
Analyzers
SPI
Decoded Protocols
Search Protocols
MOSI: 0x00; MISO: 0x9F
MOSI: 0xC2; MISO: 0x00
MOSI: 0x00; MISO: 0x05
MOSI: 0x38; MISO: 0x00
MOSI: 0x00; MISO: 0x03
MOSI: 0x00; MISO: 0x00
MOSI: 0x00; MISO: 0x00
MOSI: 0x00; MISO: 0x00
MOSI: 0xA2; MISO: 0x00
MOSI: 0x01; MISO: 0x00
MOSI: 0x00; MISO: 0x03
MOSI: 0x00; MISO: 0x3F
MOSI: 0x00; MISO: 0xE0
Capture

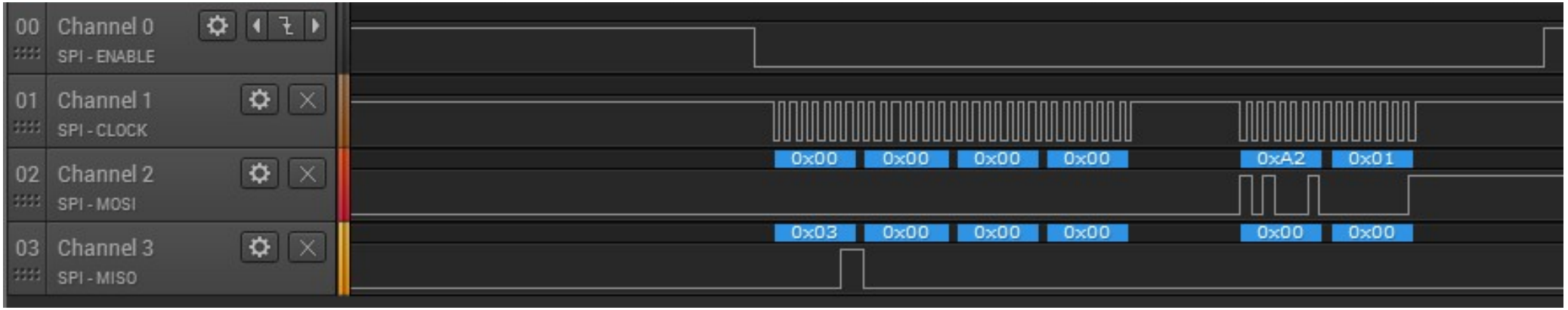

Saleae Logic 1.2.9 - [Connected] - [3 MHz Digital, 1 s]
Options
Start
0 s : 203 ms : 200 µs
0 s : 203 ms : 300 µs
+10 µs +20 µs +30 µs +40 µs +50 µs +60 µs +70 µs +80 µs +90 µs
00 Channel 0
SPI - ENABLE
01 Channel 1
SPI - CLOCK
02 Channel 2
SPI - MOSI
03 Channel 3
SPI - MISO
0x00 0x00 0x00 0x00 0xA2 0x01 0x00 0x00
0x03 0x00 0x00 0x00 0x00 0x00 0x03 0x3F
Annotations
Timing Marker Pair
| A1 - A2 | = ###
A1 @ ###
A2 @ ###
Analyzers
SPI
Decoded Protocols
Search Protocols
MOSI: 0x00; MISO: 0x9F
MOSI: 0xC2; MISO: 0x00
MOSI: 0x00; MISO: 0x05
MOSI: 0x38; MISO: 0x00
MOSI: 0x00; MISO: 0x03
MOSI: 0x00; MISO: 0x00
MOSI: 0x00; MISO: 0x00
MOSI: 0x00; MISO: 0x00
MOSI: 0xA2; MISO: 0x00
MOSI: 0x01; MISO: 0x00
MOSI: 0x00; MISO: 0x03
MOSI: 0x00; MISO: 0x3F
MOSI: 0x00; MISO: 0xE0
Capture



10. COMMAND DESCRIPTION

Table4. Command Definition

Command (byte)	WREN (write enable)	WRDI (write disable)	WRSR (write status register)	RDID (read identification)	RDSR (read status register)	READ (read data)	FAST READ (fast read data)
1st byte	06 (hex)	04 (hex)	01 (hex)	9F (hex)	05 (hex)	03 (hex)	0B (hex)
2nd byte						AD1	AD1
3rd byte						AD2	AD2
4th byte						AD3	AD3
5th byte							Dummy
Action	sets the (WEL) write enable latch bit	resets the (WEL) write enable latch bit	to write ne w values to the status register	outputs JEDEC ID: 1-byte Manufacturer ID & 2-byte Device ID	to read out the values of the status register	n bytes read out until CS# goes high	n bytes read out until CS# goes high



10. COMMAND DESCRIPTION

Table4. Command Definition

Command (byte)	WREN (write enable)	WRDI (write disable)	WRSR (write status register)	RDID (read identification)	RDSR (read status register)	READ (read data)	FAST READ (fast read data)
1st byte	06 (hex)	04 (hex)	01 (hex)	9F (hex)	05 (hex)	03 (hex)	0B (hex)
2nd byte						AD1	AD1
3rd byte						AD2	AD2
4th byte						AD3	AD3
5th byte							Dummy
Action	sets the (WEL) write enable latch bit	resets the (WEL) write enable latch bit	to write ne w values to the status register	outputs JEDEC ID: 1-byte Manufacturer ID & 2-byte Device ID	to read out the values of the status register	n bytes read out until CS# goes high	n bytes read out until CS# goes high

Closing Remarks

- Get started hardware hacking!
- Open discussion / questions
- Thank you!

  @TheDukeZip

  https://github.com/thedukezip/bsidesboston2016