



Assisted Discovery of On-Chip Debug Interfaces

Joe Grand (@joegrand)

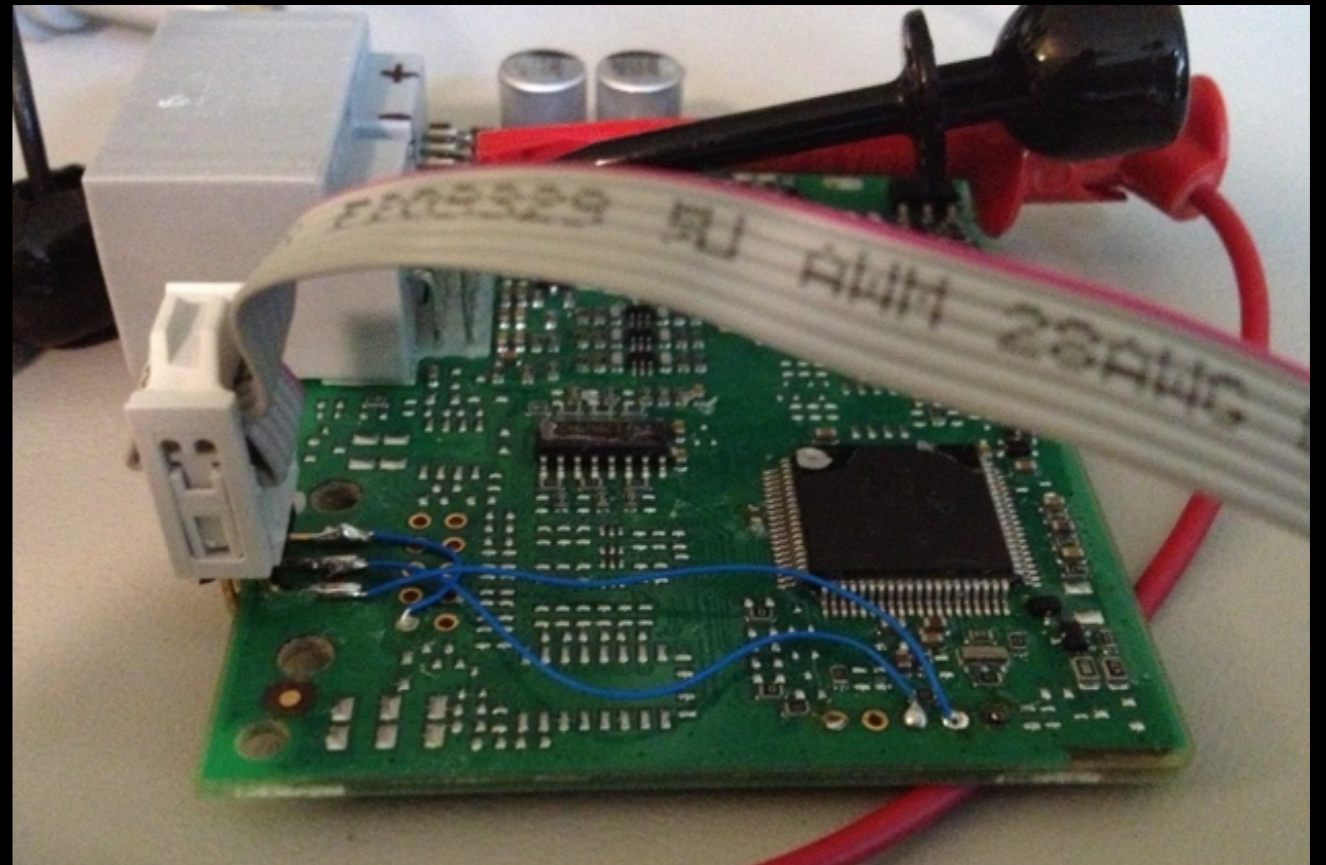
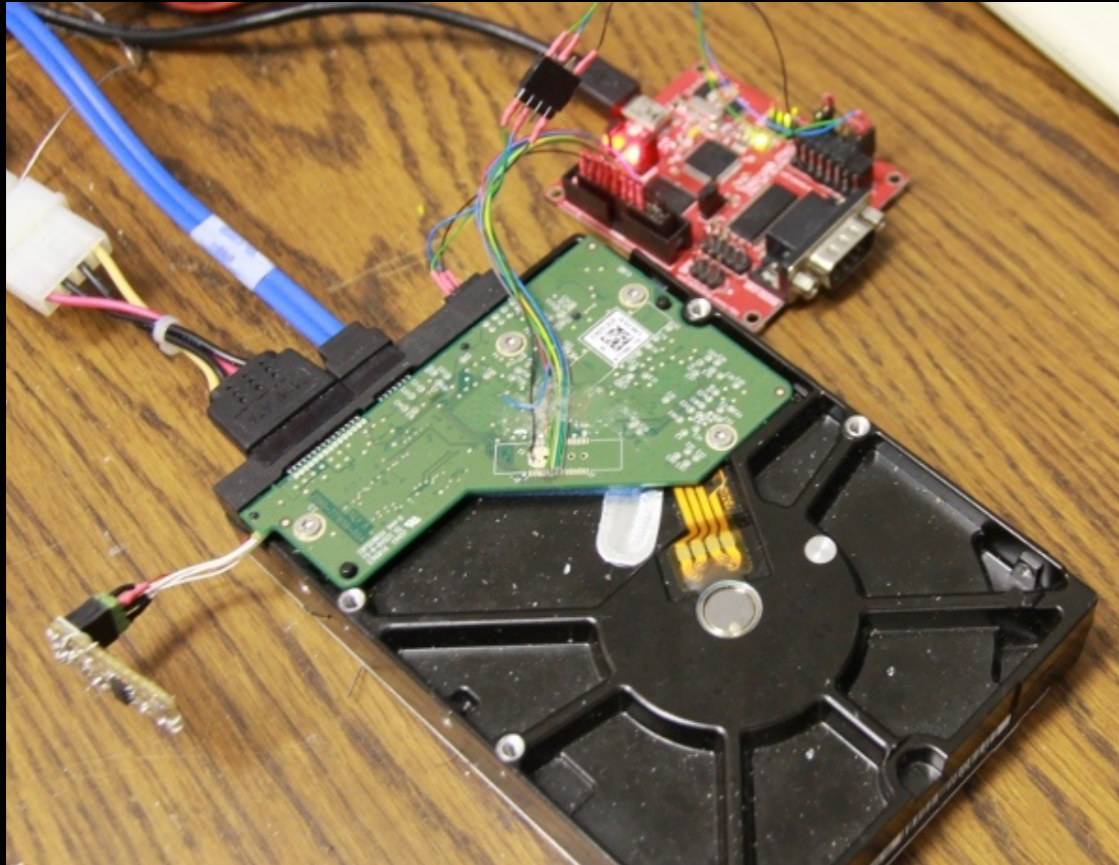
Introduction

- On-chip debug interfaces are a well-known attack vector
 - Used as a stepping stone to further an attack
 - Can provide chip-level control of a target device
 - Extract program code or data
 - Modify memory contents
 - Affect device operation on-the-fly
- Inconvenient for vendor to remove functionality
 - Would prevent capability for legitimate personnel
 - Obfuscated or password protected instead



Introduction 2

- Identifying OCD interfaces can sometimes be difficult and/or time consuming

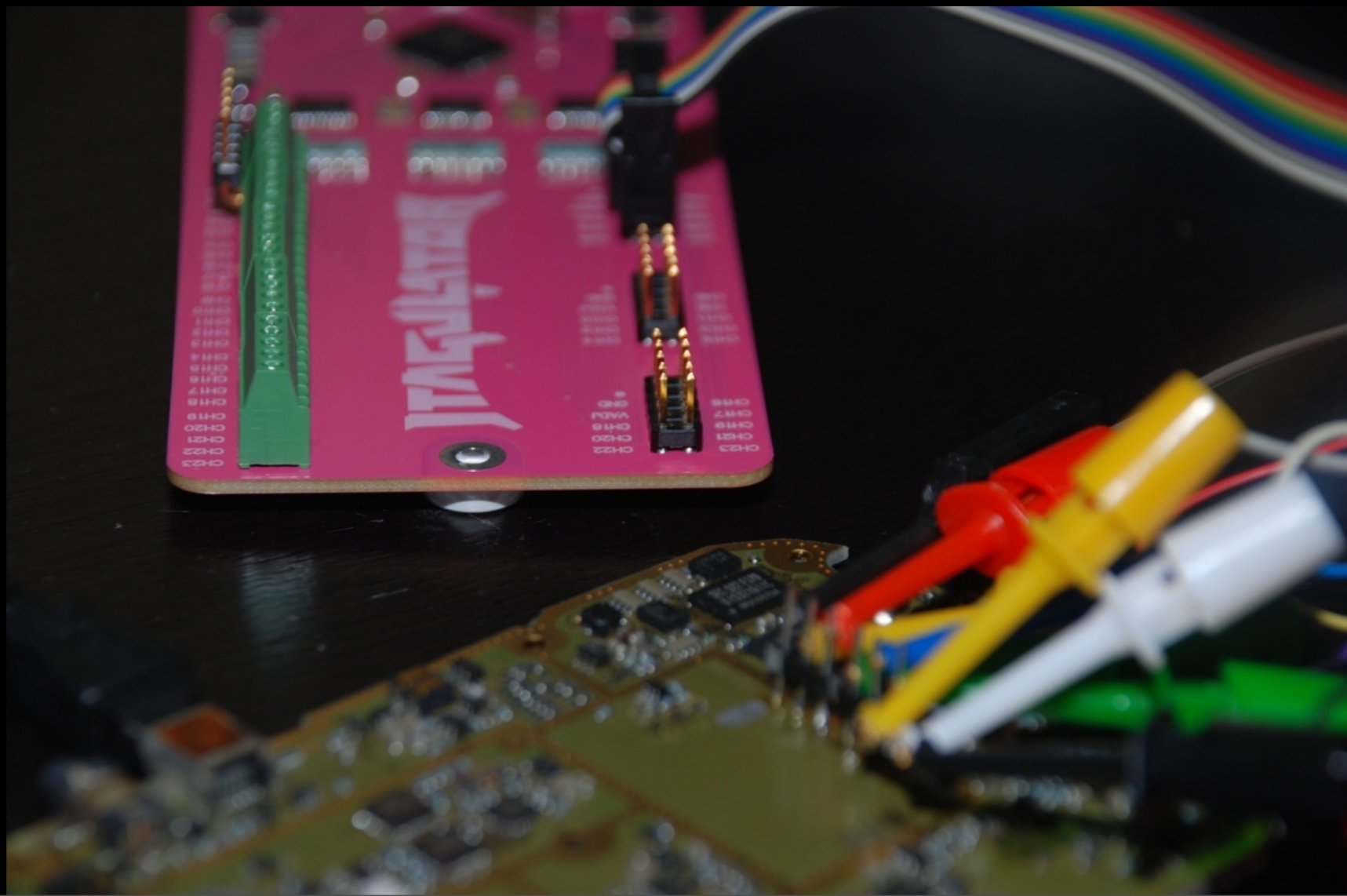


← <http://spritesmods.com/?art=hddhack>



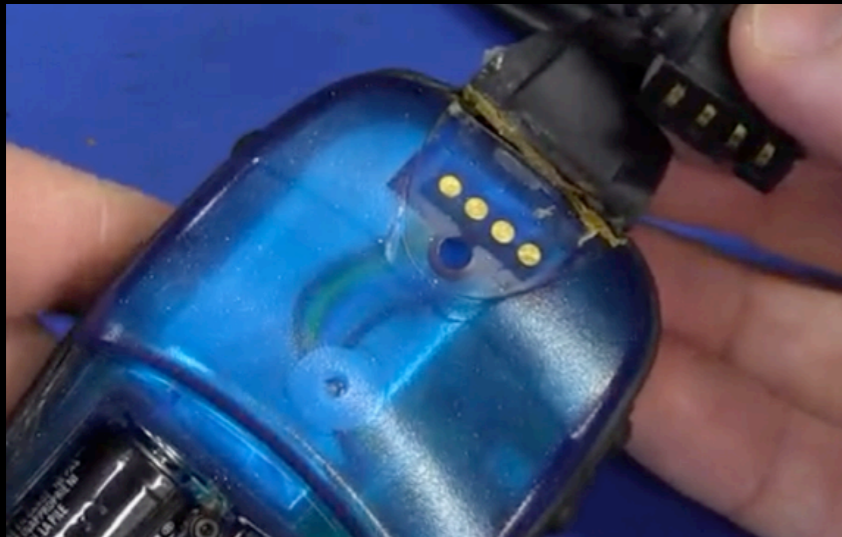
Goals

- Create an easy-to-use tool to simplify the process
- Attract non-HW folks to HW hacking



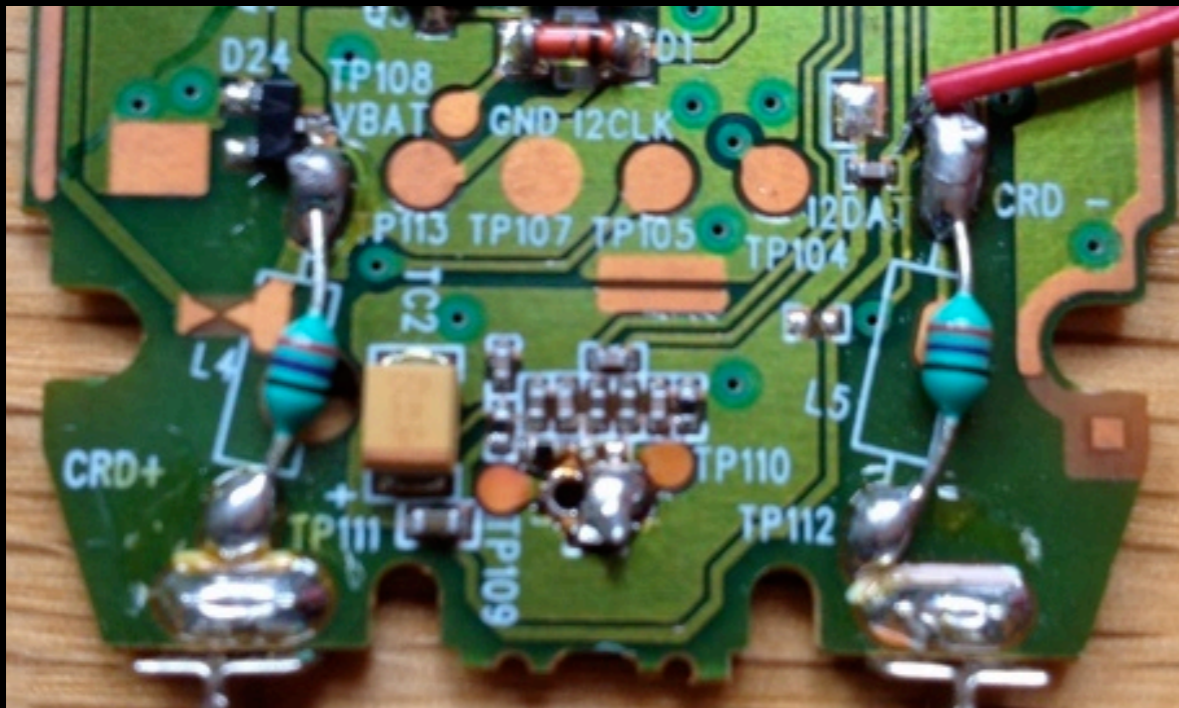
Identifying Interfaces: External

- Accessible to the outside world
 - Intended for engineers or manufacturers
 - Device programming or final system test
- Usually hidden or protected
 - Underneath batteries
 - Behind stickers/covers
- May be a proprietary/non-standard connector



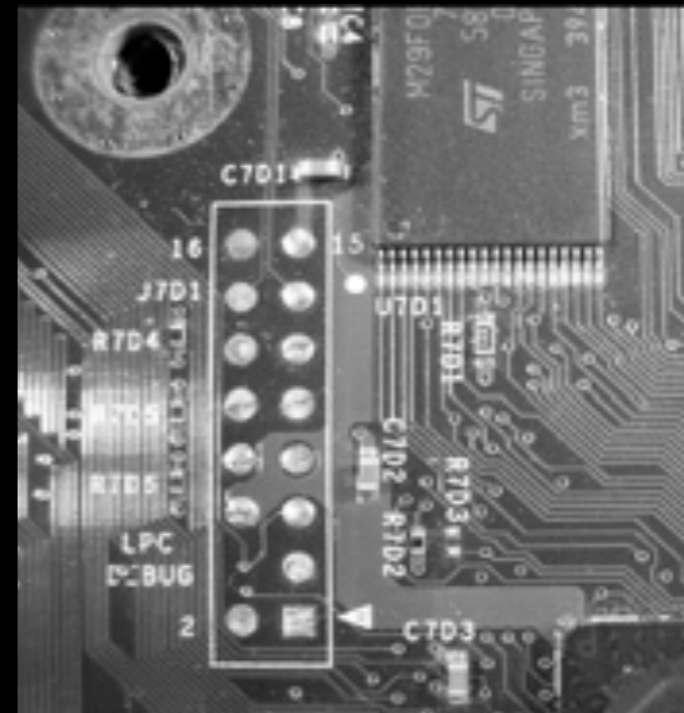
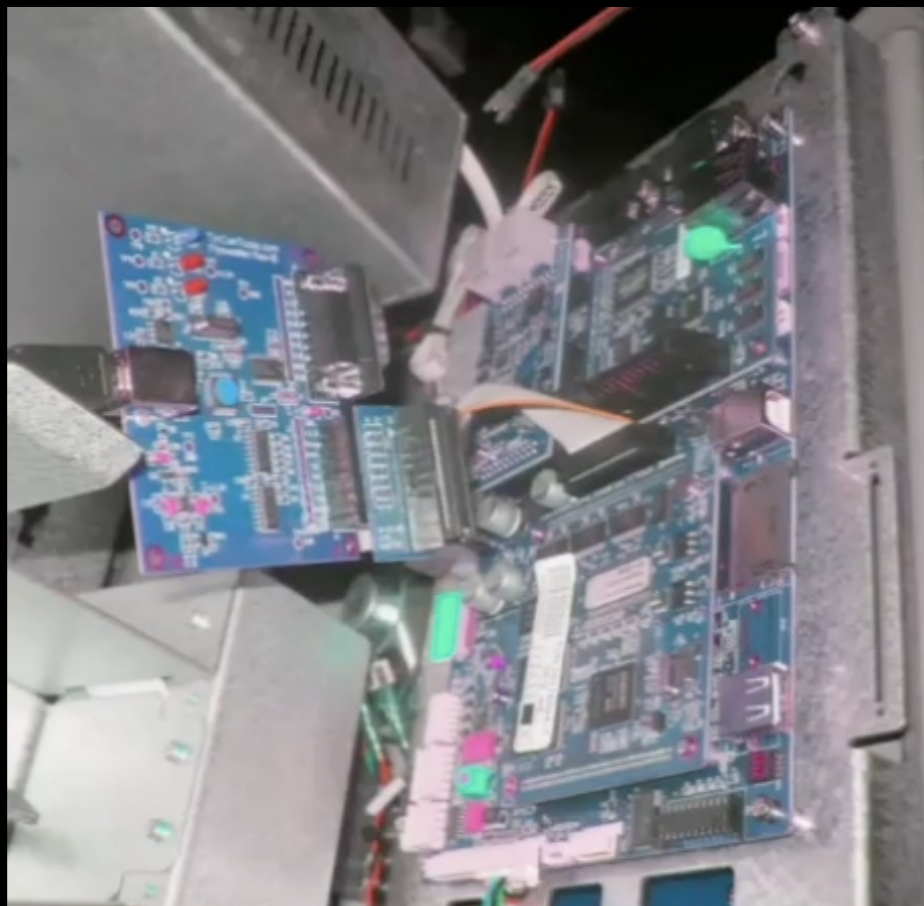
Identifying Interfaces: Internal

- Test points or unpopulated pads
- Silkscreen markings or notation
- Easy-to-access locations



Identifying Interfaces: Internal 2

- Familiar target or based on common pinout
 - Often single- or double-row footprint
 - JTAG: www.jtagtest.com/pinouts/

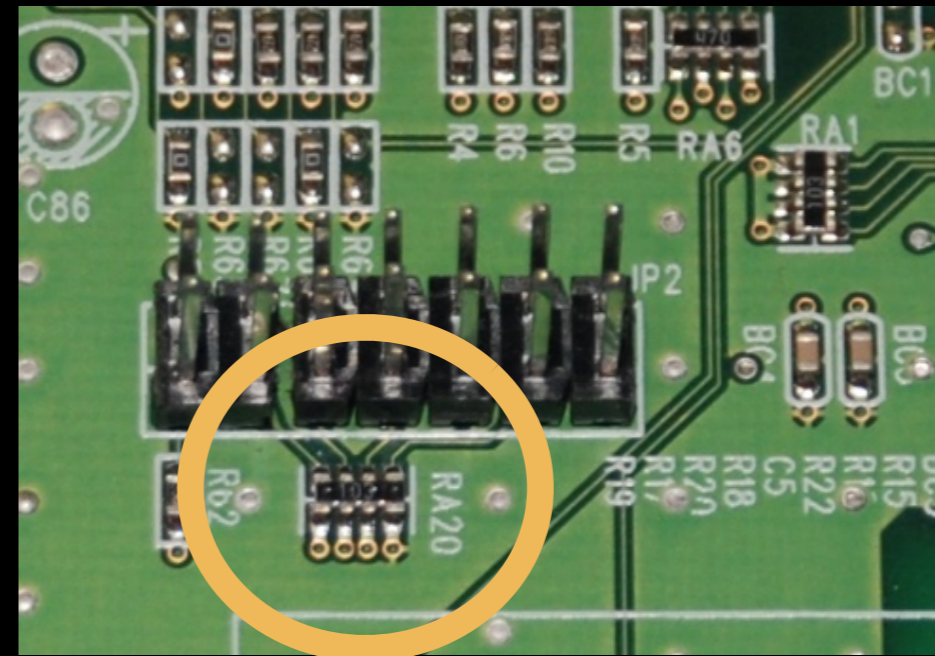
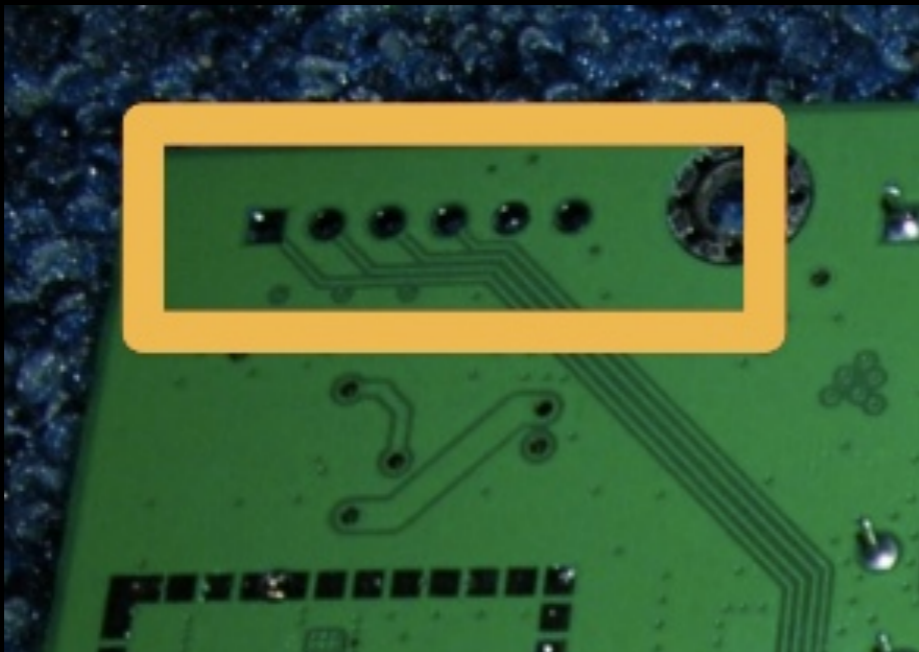


- ← www.blackhat.com/html/bh-us-10/bh-us-10-archives.html#jack
- www.nostarch.com/xboxfree



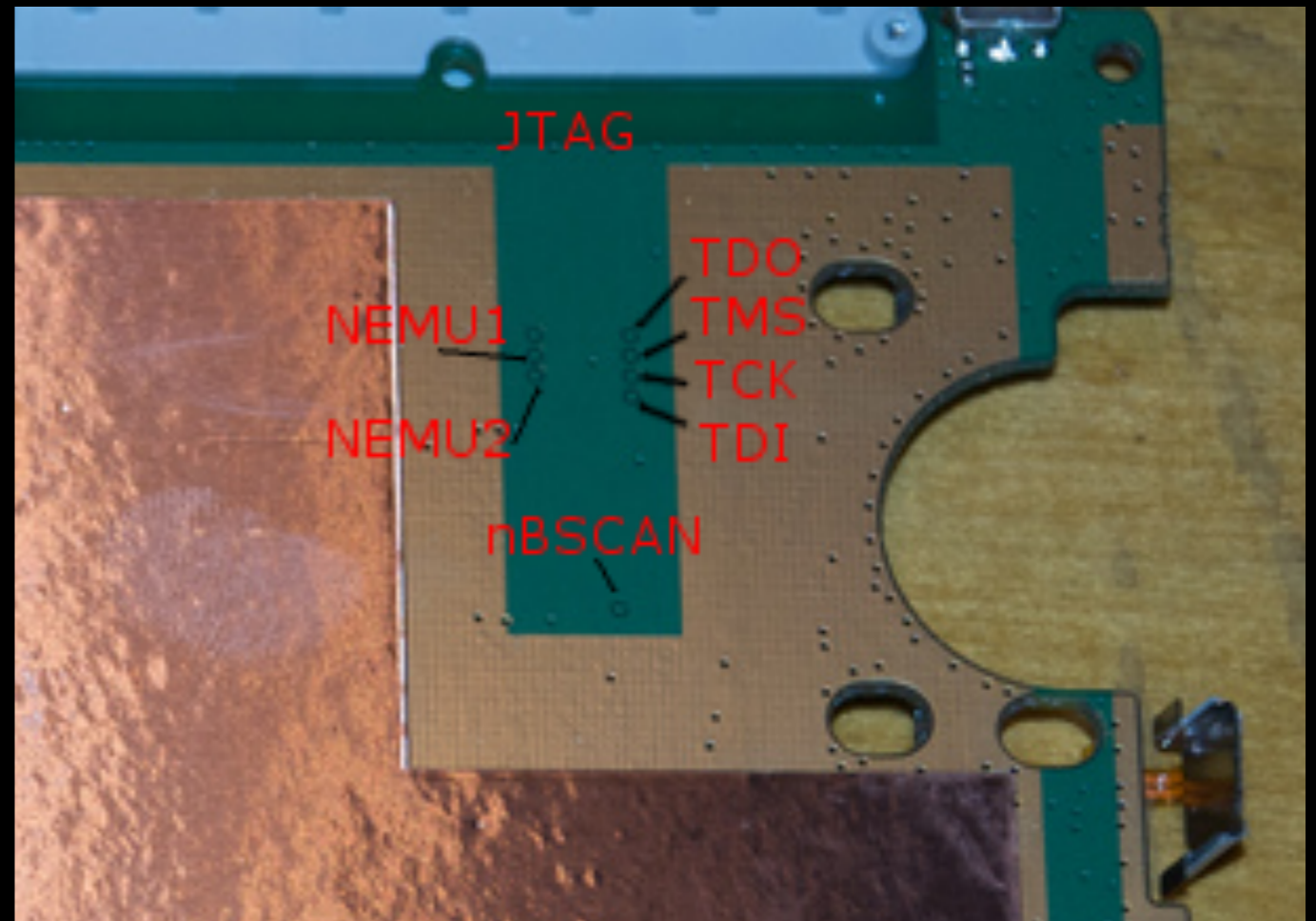
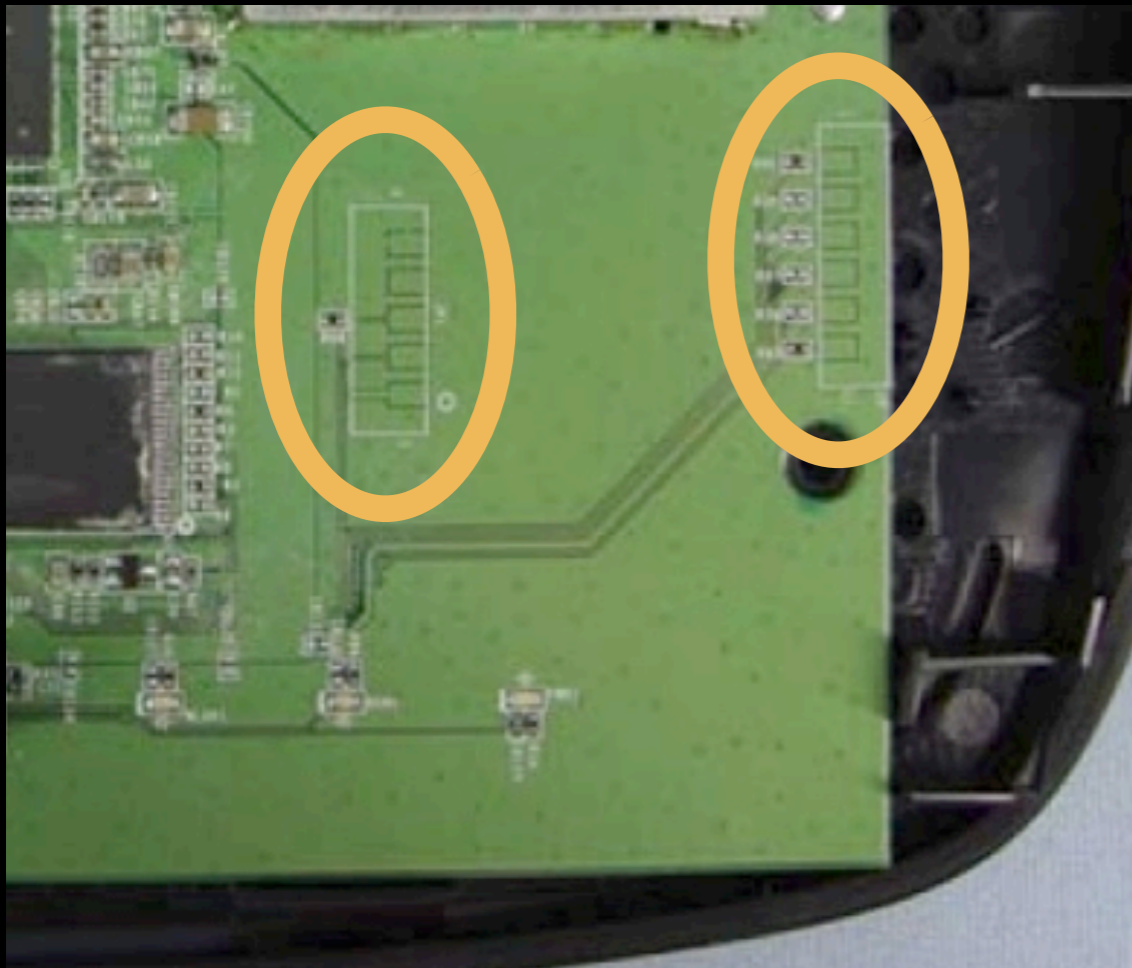
Identifying Interfaces: Internal 3

- Can use PCB/design heuristics
 - Traces of similar function are grouped together (bus)
 - Array of pull-up/pull-down resistors (to set static state of pins)
 - Test points usually placed on important/interesting signals



Identifying Interfaces: Internal 4

- Might be covered by soldermask



← Linksys WRT54G2 v1.3

→ <http://elinux.org/File:Peekjtag3.png>



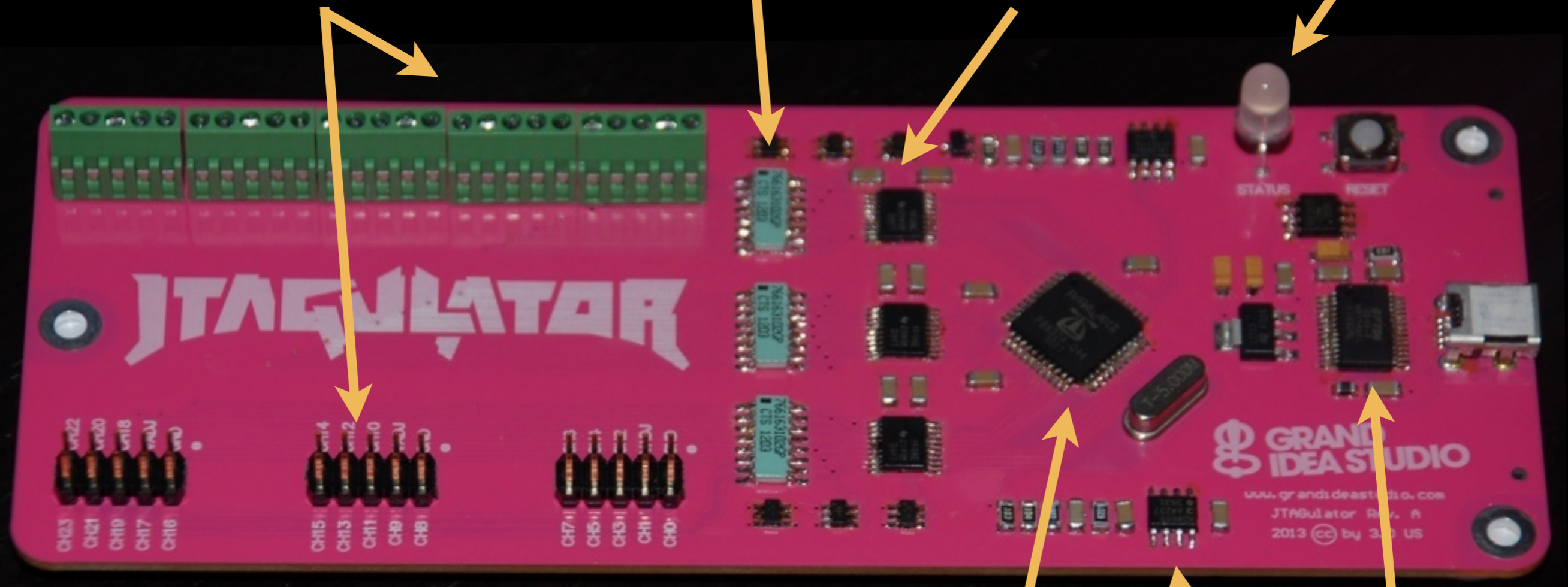
PCB

Input protection

Target I/F (24 channels)

Level translation

Status



*** 2x5 headers compatible w/ Bus Pirate probes,
http://dangerousprototypes.com/docs/Bus_Pirate

Propeller

Op-Amp/DAC

USB



Propeller/Core



- Completely custom, ground up, open source
- 8 parallel 32-bit processors (cogs)
- Code in Spin, ASM, or C



*** INFORMATION: www.parallax.com/propeller/

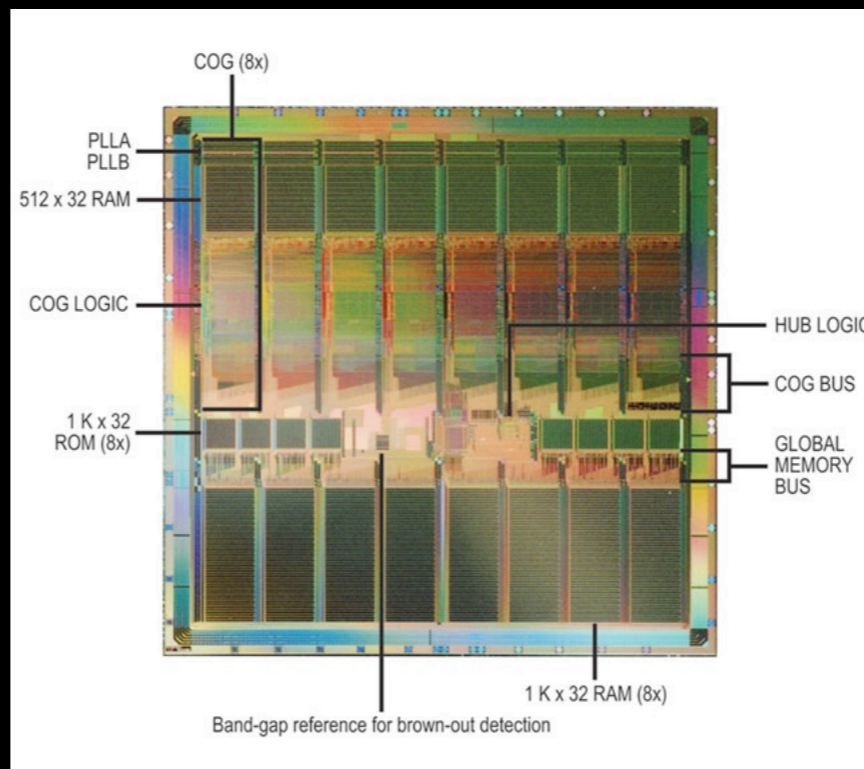
*** DISCUSSION FORUMS: <http://forums.parallax.com>

*** OBJECT EXCHANGE: <http://obex.parallax.com>



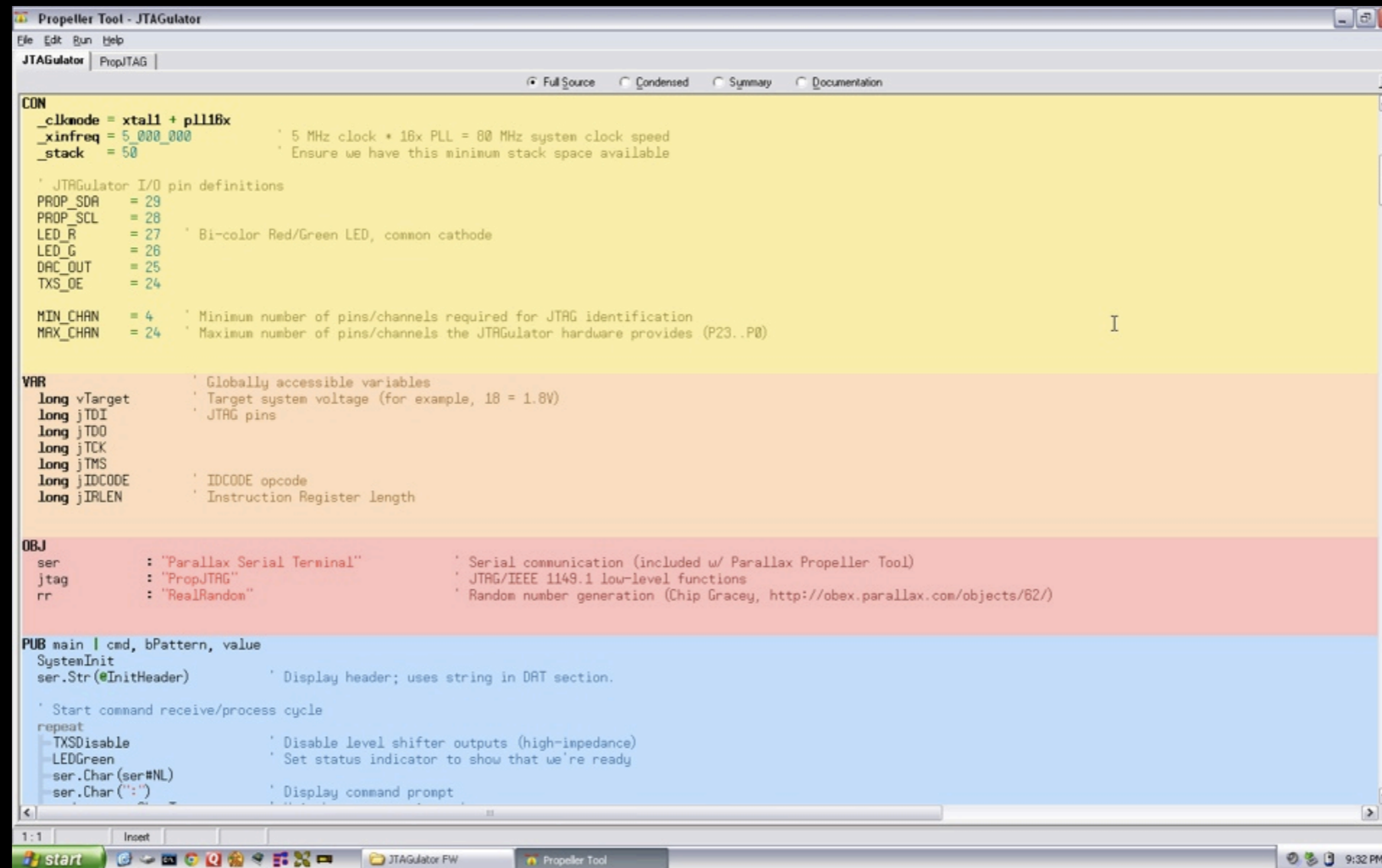
Propeller/Core 2

- Clock: DC to 128MHz (80MHz recommended)
- Global (hub) memory: 32KB RAM, 32KB ROM
- Cog memory: 2KB RAM each
- GPIO: 32 @ 40mA sink/source per pin
- Program code loaded from external EEPROM on power-up



Propeller/Core 3

- Standard development using Propeller Tool & Parallax Serial Terminal (Windows)
- Programmable via serial interface (usually in conjunction w/ USB-to-serial IC)



```
Propeller Tool - JTAGulator
File Edit Run Help
JTAGulator PropJTAG
Full Source Condensed Summary Documentation

CON
_clknode = xtall + pll18x
_xinfreq = 5_000_000      * 5 MHz clock * 18x PLL = 90 MHz system clock speed
_stack = 50              * Ensure we have this minimum stack space available

* JTAGulator I/O pin definitions
PROP_SDA = 29
PROP_SCL = 28
LED_R = 27 * Bi-color Red/Green LED, common cathode
LED_G = 26
DAC_OUT = 25
TXS_OE = 24

MIN_CHAN = 4 * Minimum number of pins/channels required for JTAG identification
MAX_CHAN = 24 * Maximum number of pins/channels the JTAGulator hardware provides (P23..P0)

VAR
* Globally accessible variables
long vTarget * Target system voltage (for example, 1.8 = 1.8V)
long jTDI * JTAG pins
long jTDO
long jTCK
long jTMS
long jIDCODE * IDCODE opcode
long jIRLEN * Instruction Register length

OBJ
ser : "Parallax Serial Terminal" * Serial communication (included w/ Parallax Propeller Tool)
jtag : "PropJTAG" * JTAG/IEEE 1149.1 low-level functions
rr : "RealRandom" * Random number generation (Chip Gracey, http://obex.parallax.com/objects/62/)

PUB main | cmd, bPattern, value
SystemInit
ser.Str(@InitHeader) * Display header; uses string in DAT section.

* Start command receive/process cycle
repeat
TXSDisable * Disable level shifter outputs (high-impedance)
LEDGreen * Set status indicator to show that we're ready
ser.Char(ser#NL)
ser.Char(":") * Display command prompt
```



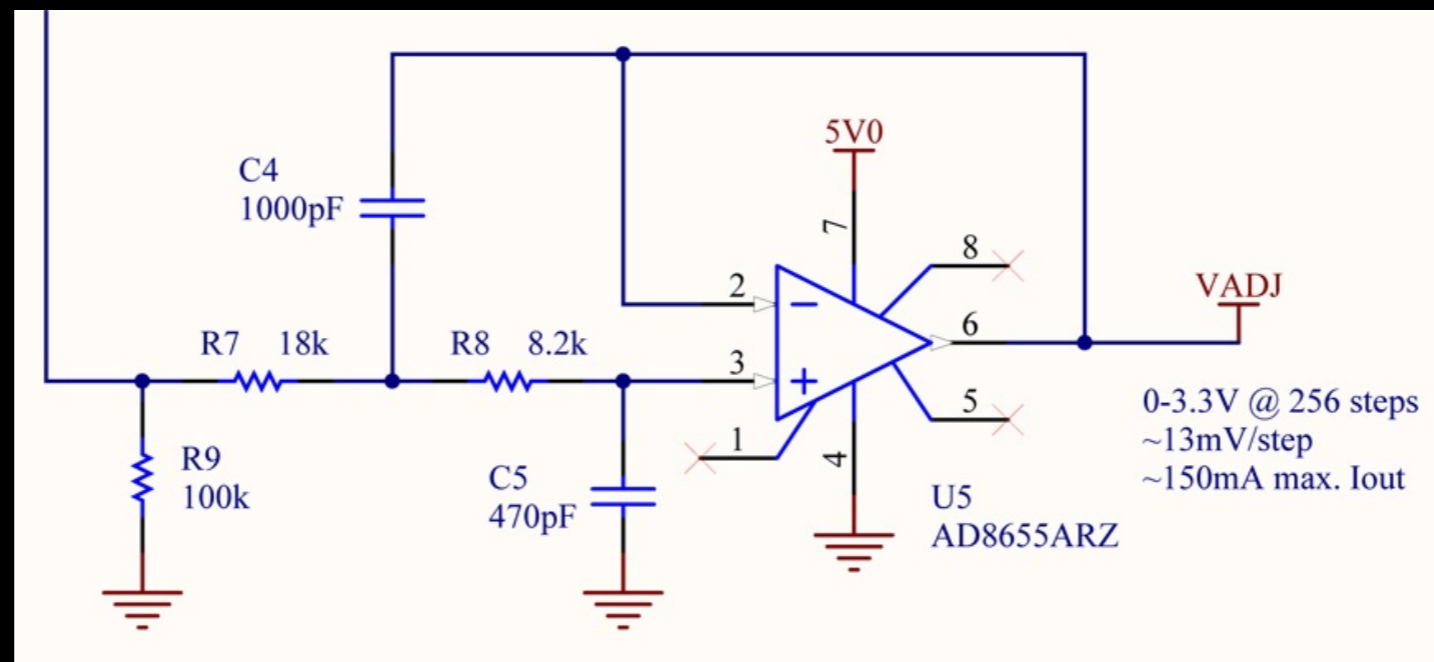
USB Interface

- Allows for Propeller programming & UI
- Powers JTAGulator from bus (5V)
- FT232RL USB-to-Serial UART
 - Entire USB protocol handled on-chip
 - Host will recognize as a virtual serial port (Windows, OS X, Linux)
- MIC2025 Power Distribution Switch
 - Internal current limiting, thermal shutdown
 - Let the FT232 enumerate first (@ < 100mA), then enable system load



Adjustable Target Voltage (VADJ)

- PWM from Propeller
 - Duty cycle corresponds to output voltage
 - Look-up table in 0.1V increments (1.2V–3.3V)
- AD8655 Low Noise, Precision CMOS Amplifier
 - Single supply, rail-to-rail
 - Voltage follower configuration
 - ~150mA output current @ $V_o = 1.2V-3.3V$



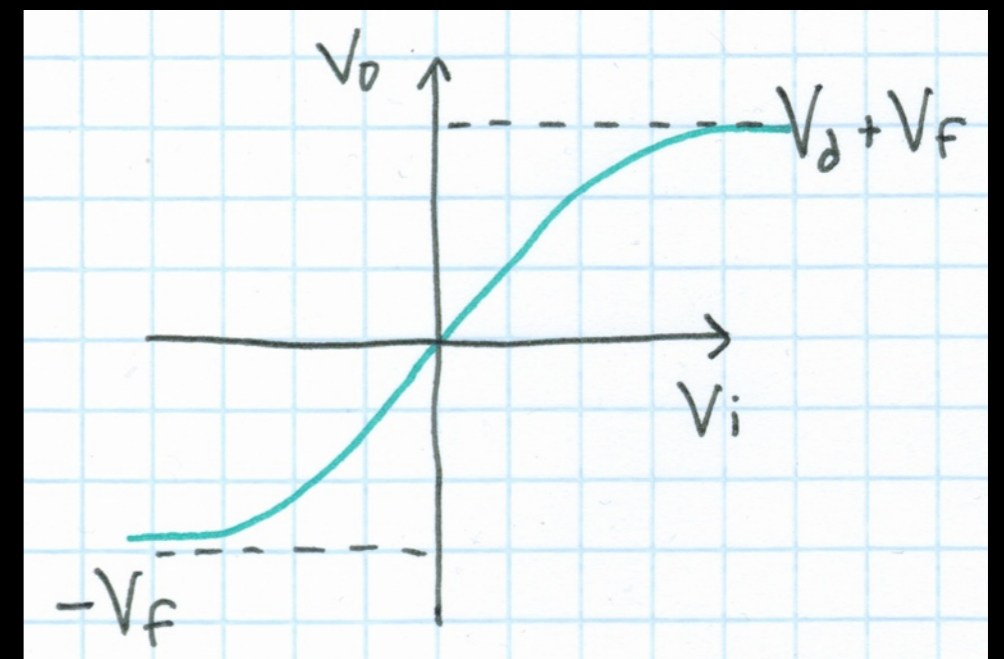
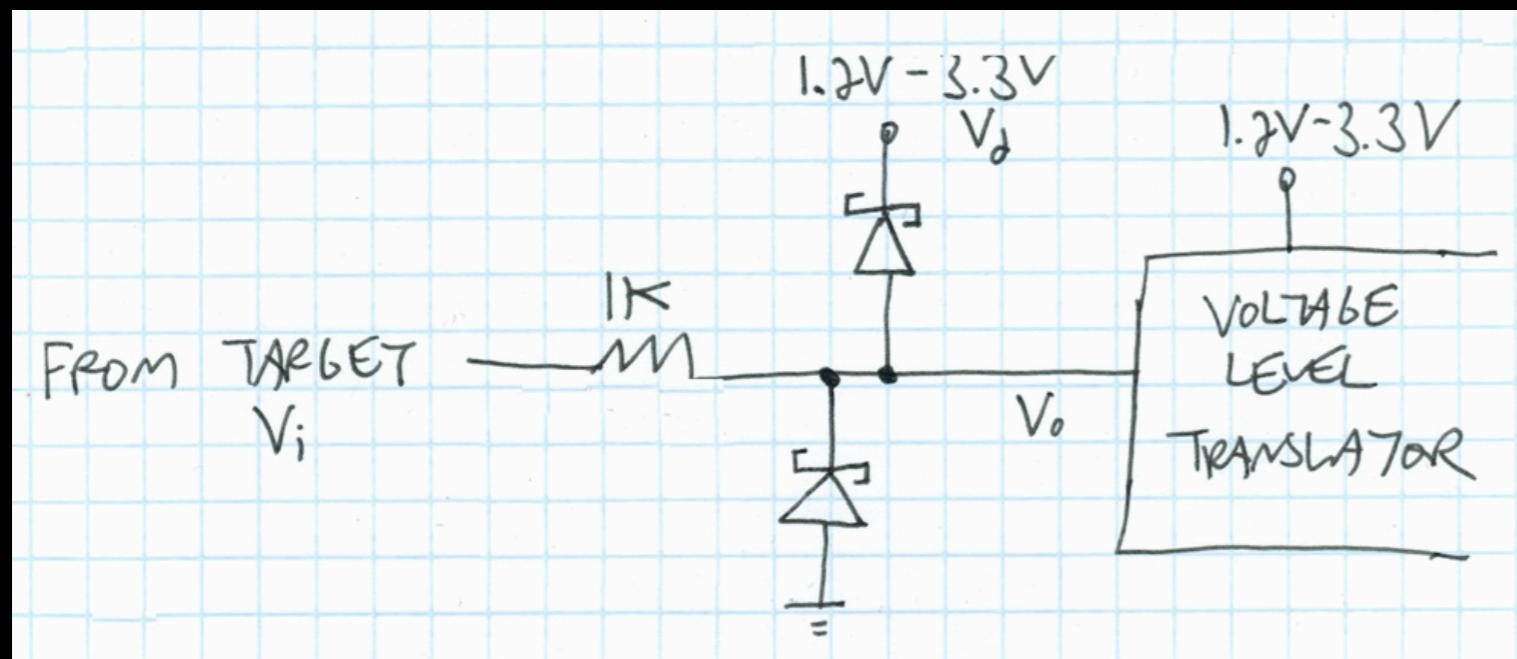
Level Translation

- Allows 3.3V signals from Propeller to be converted to VADJ
- Prevents potential damage due to over-voltage on target device's unknown connections
- TXS0108E Bidirectional Voltage-Level Translator
 - Designed for both open drain and push-pull interfaces
 - Internal pull-up resistors (40k Ω when driving low, 4k Ω when high)
 - Automatic signal direction detection
 - High-Z outputs when OE low \rightarrow will not interfere with target when not in use



Input Protection

- Prevent high voltages/spikes on unknown pins from damaging JTAGulator
- Diode limiter clamps input if needed
- V_f must be $< 0.5V$ to protect TXS0108Es



On-Chip Debug Interfaces

- JTAG
- UART



JTAG

- **Industry-standard interface (IEEE 1149.1)**
 - Created for chip- and system-level testing
 - Defines low-level functionality of finite state machine/ Test Access Port (TAP)
 - http://en.wikipedia.org/wiki/Joint_Test_Action_Group
- **Provides a direct interface to hardware**
 - Can "hijack" all pins on the device (Boundary scan/test)
 - Can access other devices connected to target chip
 - Programming/debug interface (access to Flash, RAM)
 - Vendor-defined functions/test modes might be available

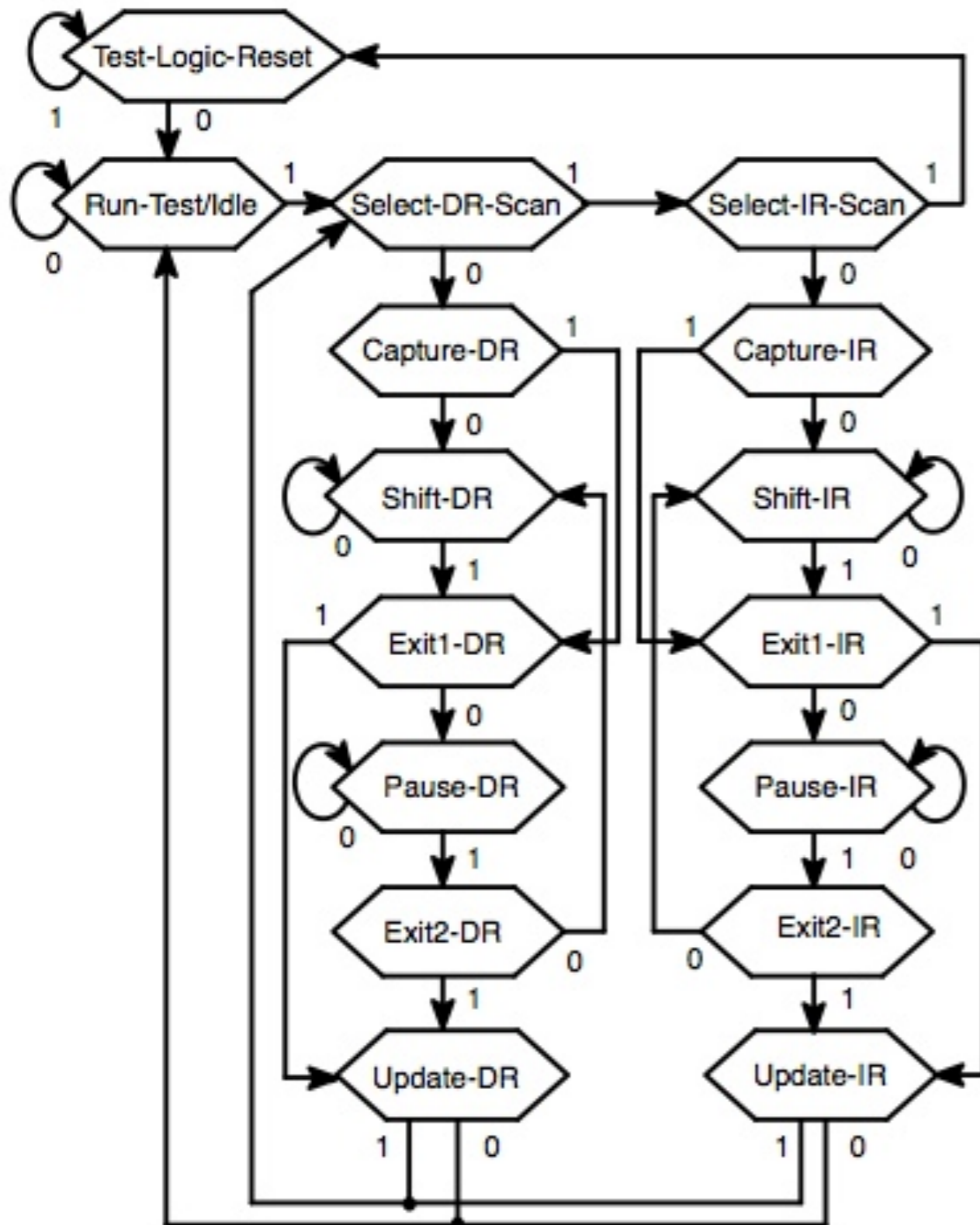


JTAG: Architecture

- Synchronous serial interface
 - TDI = Data In (to target device)
 - ← TDO = Data Out (from target device)
 - TMS = Test Mode Select
 - TCK = Test Clock
 - /TRST = Test Reset (optional for async reset)
- Test Access Port (TAP) w/ Shift Registers
 - Instruction (≥ 2 bit wide)
 - Data
 - Bypass (1 bit)
 - Boundary Scan (variable)
 - Device ID (32 bit) (optional)



JTAG:TAP Controller



*** State transitions occur on rising edge of TCK based on current state and value of TMS

*** TAP provides 4 major operations: Reset, Run-Test, Scan DR, Scan IR

*** Can move to Reset state from any other state w/ TMS high for 5x TCK

*** 3 primary steps in Scan: Capture, Shift, Update

*** Data held in "shadow" latch until Update state



JTAG: Protection

- Implementation specific
- Security fuse physically blown prior to release
 - Could be repaired w/ silicon die attack
- Password required to enable functionality
 - Ex.: Flash erased after n attempts (so perform $n-1$), then reset and continue
- May allow BYPASS, but prevent higher level functionality
 - Ex.: TI MSP430



JTAG: HW Tools

- **RIFF Box**
 - www.jtagbox.com
- **H-JTAG**
 - www.hjtag.com/en/
- **SEGGER J-Link**
 - www.segger.com/debug-probes.html
- **Bus Blaster (open source)**
 - http://dangerousprototypes.com/docs/Bus_Blaster
- **Wiggler or compatible (parallel port)**
 - ftp://www.keith-koep.com/pub/arm-tools/jtag/jtag05_sch.pdf



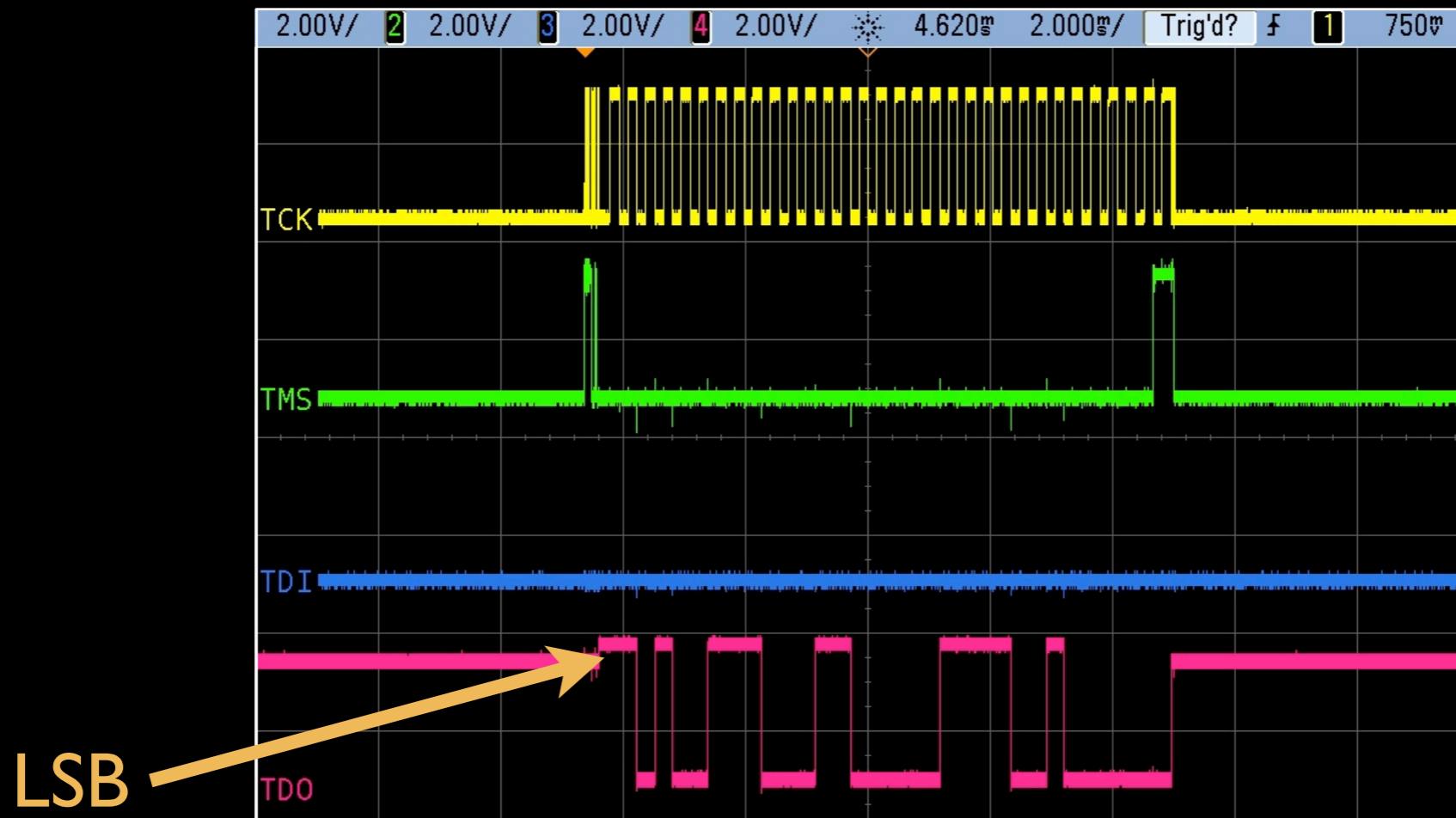
JTAG: SW Tools

- OpenOCD (Open On-Chip Debugger)
 - <http://openocd.sourceforge.net>
- UrJTAG (Universal JTAG Library)
 - www.urjtag.org



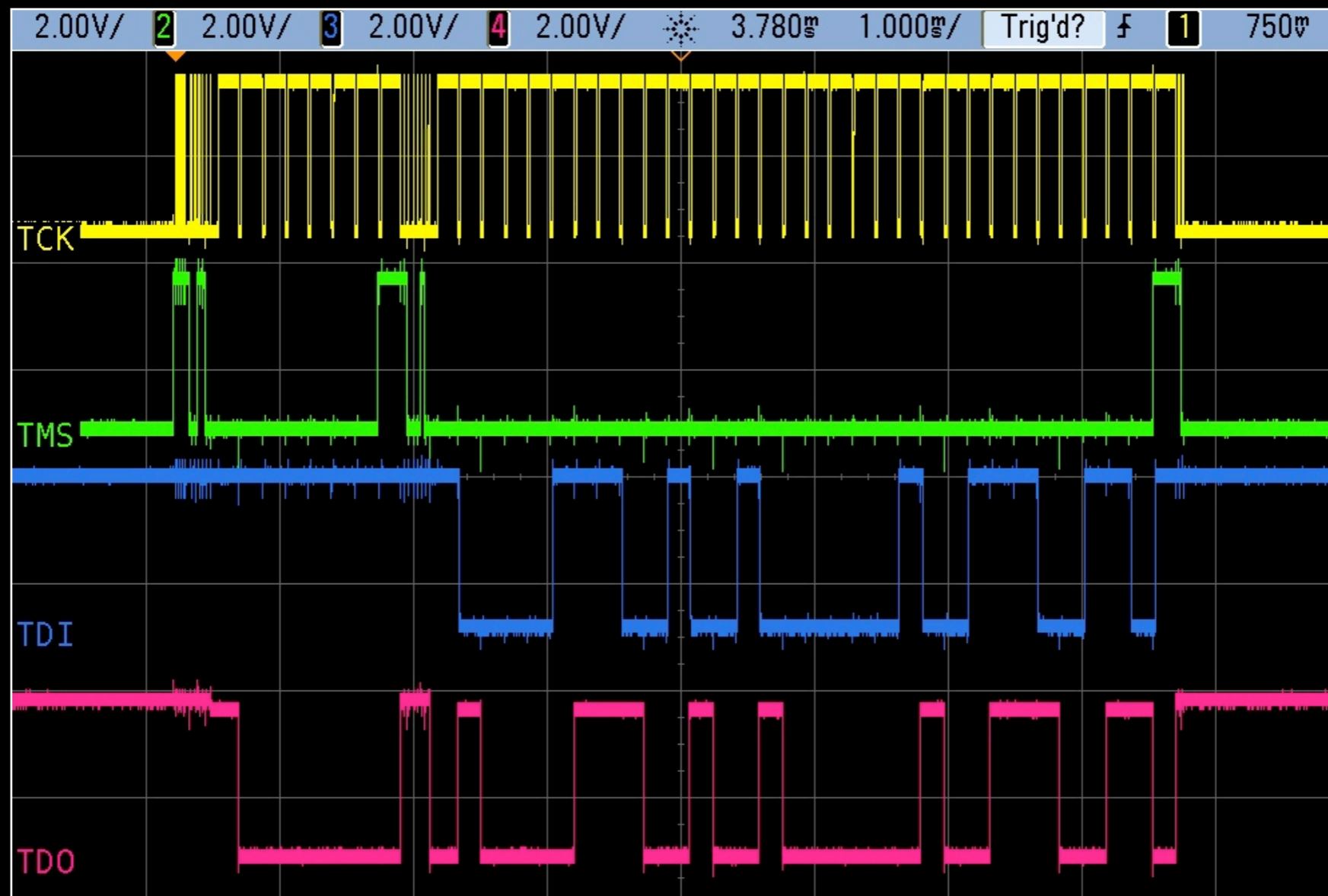
IDCODE Scan

- 32-bit Device ID (if available) is in the DR on TAP reset or IC power-up
 - Otherwise, TAP will reset to BYPASS (LSB = 0)
 - Can simply enter Shift-DR state and clock out on TDO
 - TDI not required/used during IDCODE acquisition



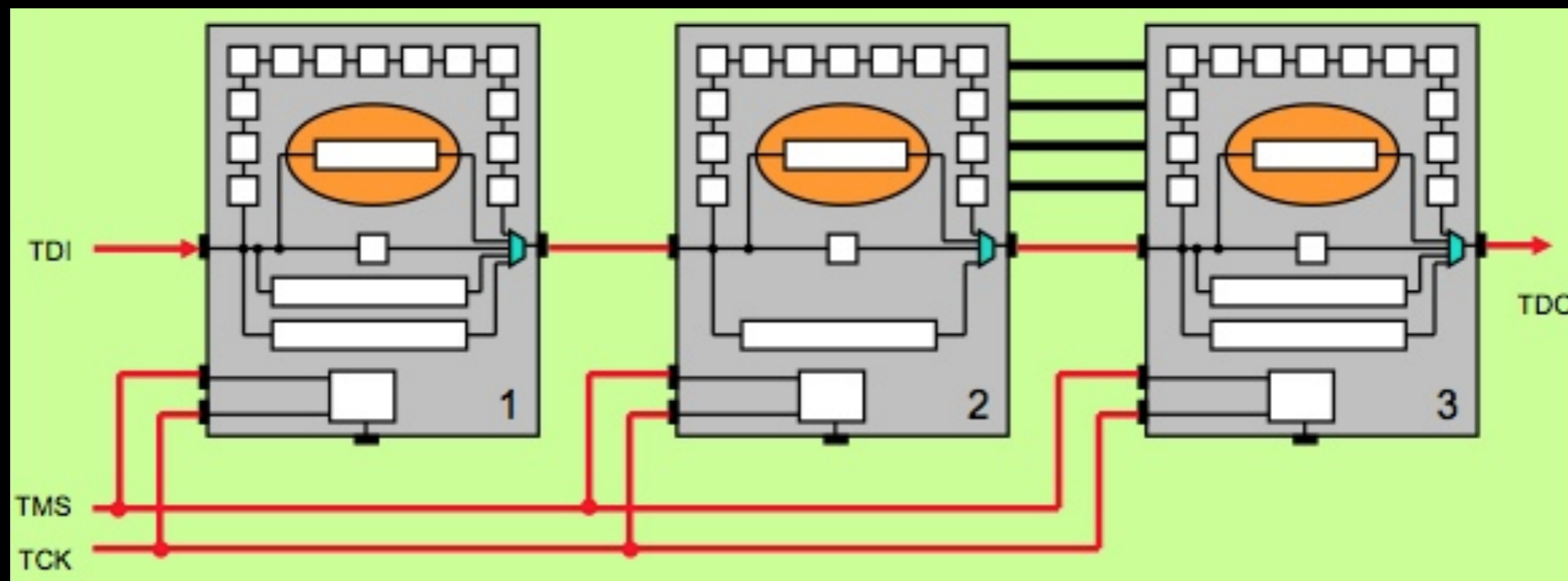
BYPASS Scan

- In BYPASS, data shifted into TDI is received on TDO delayed by one clock cycle



BYPASS Scan 2

- Can determine how many devices (if any) are in the chain via "blind interrogation"
 - Force device(s) into BYPASS (IR of all 1s)
 - Send 1s to fill DRs
 - Send a 0 and count until it is output on TDO



UART

- Universal Asynchronous Receiver/Transmitter
 - No external clock needed
 - Data bits sent LSB first (D0)
 - NRZ (Non-Return-To-Zero) coding
 - Transfer speed (bits/second) = 1 / bit width
 - http://en.wikipedia.org/wiki/Asynchronous_serial_communication

1	2	3	4	5	6	7	8	9	10	11
Start bit	5-8 data bits							Stop bit(s)		
Start	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Stop	

*** Start bit + Data bits + Parity (optional) + Stop bit(s)



UART 2

- **Asynchronous serial interface**
 - TXD = Transmit data (to target device)
 - ← RXD = Receive data (from target device)
 - ↔ DTR, DSR, RTS, CTS, RI, DCD = Control signals
(uncommon for modern implementations)
- **Many embedded systems use UART as debug output/console/root shell**

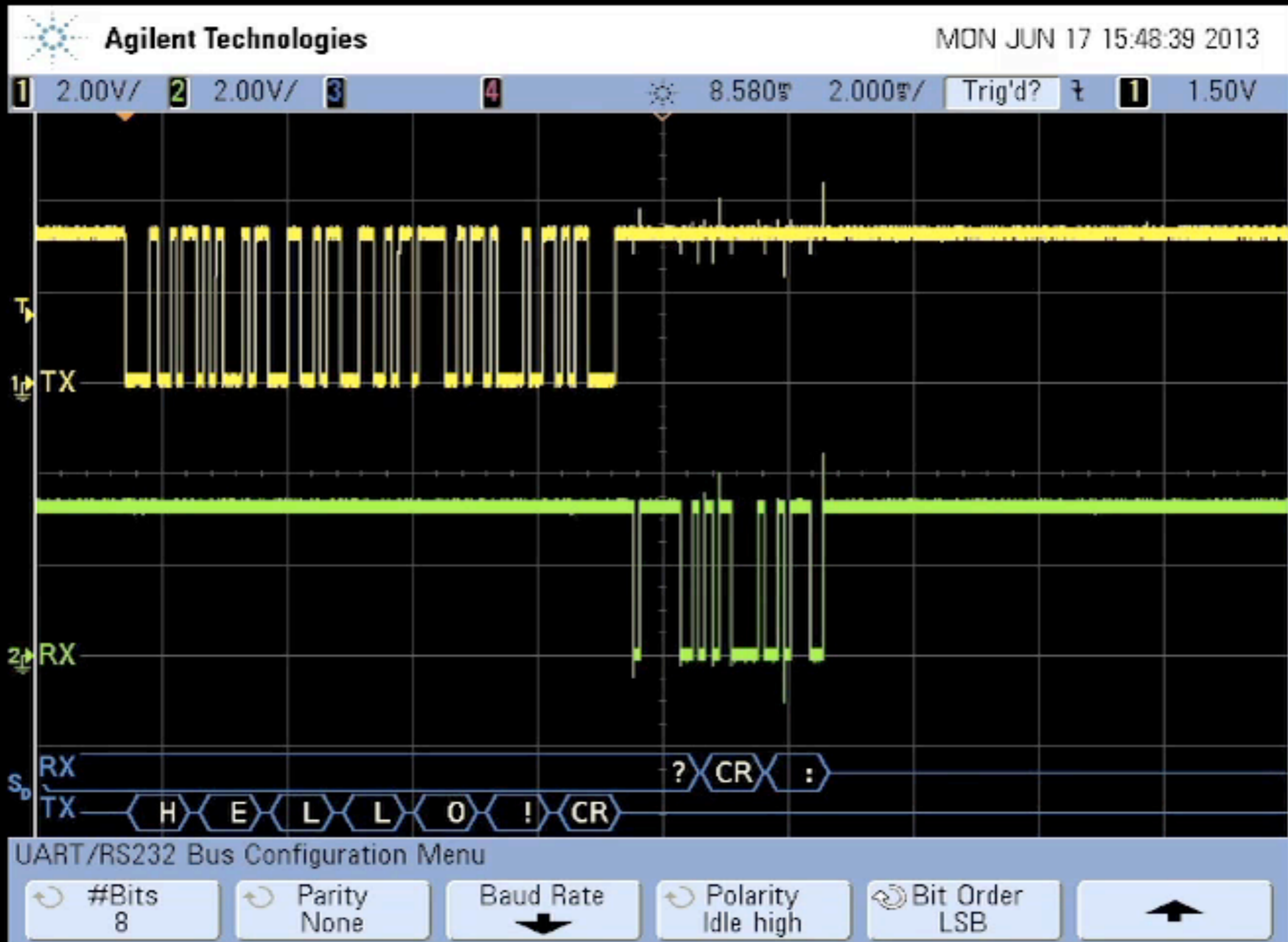


UART Scan

- 8 data bits, no parity, 1 stop bit (8N1)
- Baud rates stored in look-up table
 - 75, 110, 150, 300, 900, 1200, 1800, 2400, 3600, 4800, 7200, 9600, 14400, 19200, 28800, 31250, 38400, 57600, 76800, 115200, 153600, 230400, 250000, 307200



UART Scan 3



Possible Limitations

- No OCD interface exists
- OCD interface is physically disconnected
 - Cut traces, missing jumpers/0 ohm resistors
- OCD interface isn't being properly enabled
 - System requires other pin settings
 - Non-standard configuration
 - Password protected
- Strong pull resistors on target prevent JTAGulator from setting/receiving proper logic levels
- Could cause target to behave abnormally due to "fuzzing" unknown pins

*** Additional reverse engineering will be necessary



Future Work

- **Support for other interfaces**
 - TI Spy-Bi-Wire, ARM Serial Wire Debug, Microchip ICSP, Atmel AVR ISP, Freescale BDM, LPC Bus, Flash memory (SPI NOR/eMMC NAND)
- **Level-shifting module?**
 - Target voltage > 5V for industrial/SCADA equipment
- **Logic analyzer?**
 - Interface w/ sigrok



Get It

- www.jtagulator.com
*** Schematics, source code, BOM, block diagram, Gerber plots, photos, videos, other documentation
- www.parallax.com
*** Assembled units, accessories
- <http://oshpark.com/profiles/joegrand>
*** Bare boards



Demonstration

