



Assisted Discovery of On-Chip Debug Interfaces

Joe Grand (@joegrant)

# Agenda

- Introduction
- Inspiration / Other Art
- Identifying Interfaces
- Design Requirements
- Hardware
- Firmware
- On-Chip Debug Interfaces
- Examples / Demonstration
- Limitations
- Future Work



# 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



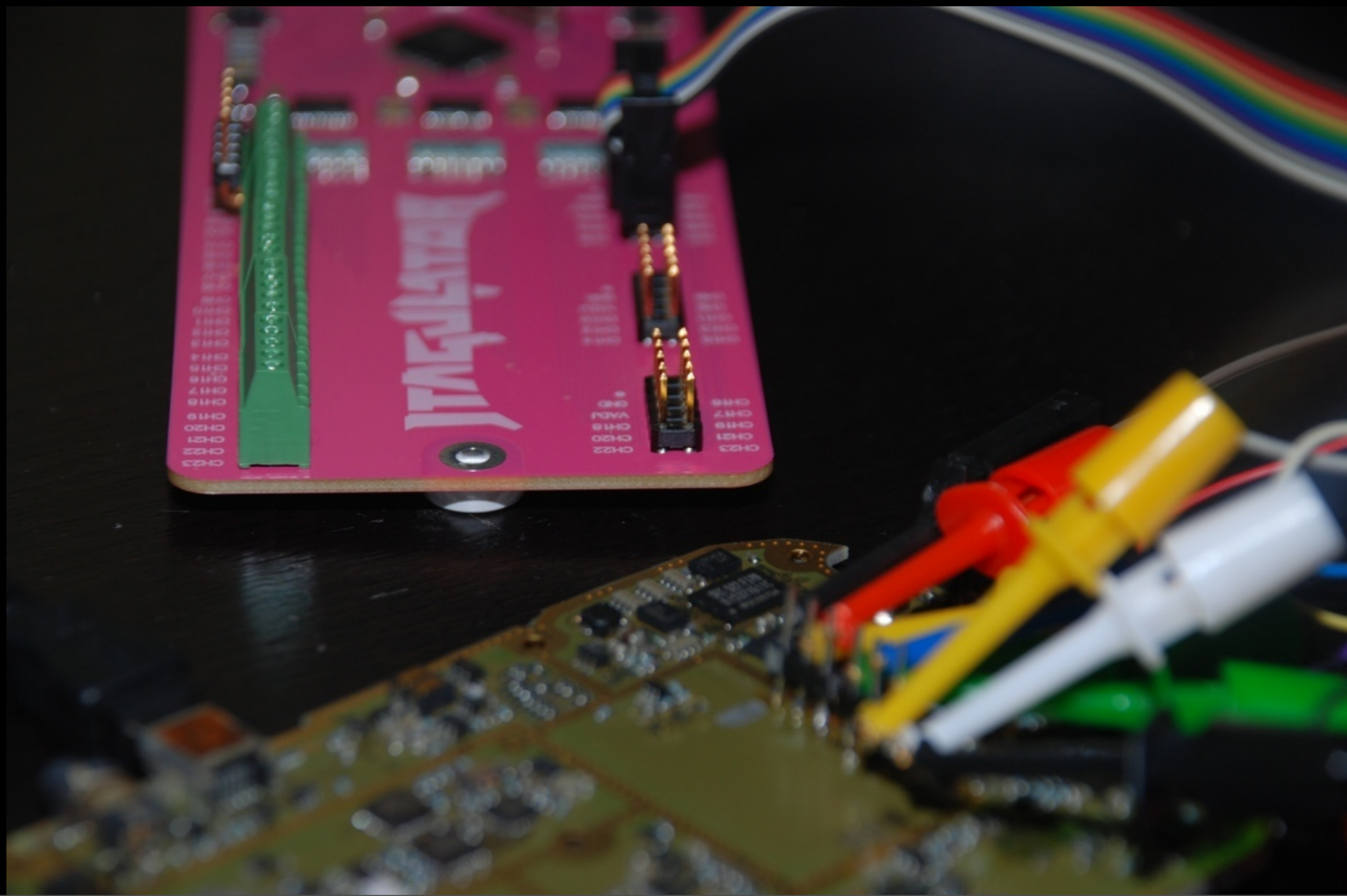
# Introduction 2

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



# Goals

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



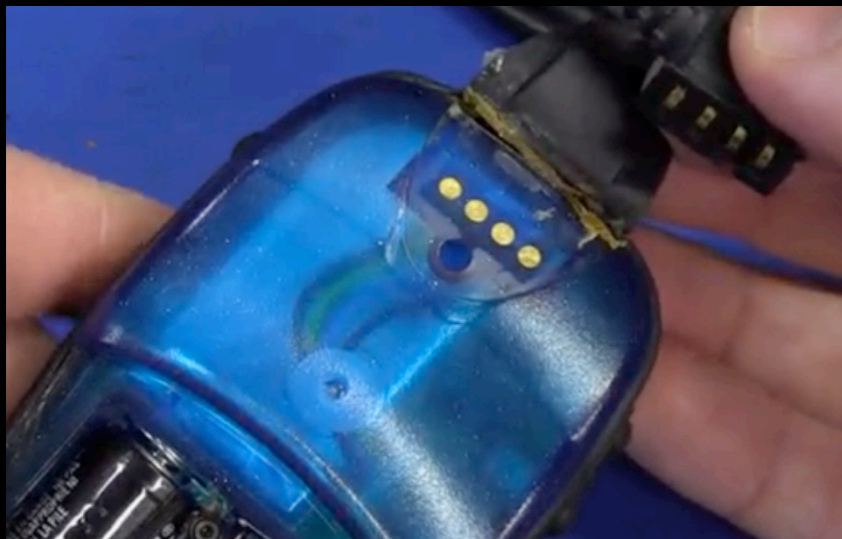
# Inspiration

- Hunz's JTAG Finder
  - [http://elinux.org/JTAG\\_Finder](http://elinux.org/JTAG_Finder)
- JTAGenum & RS232enum
  - <http://deadhacker.com/tools/>
- DARPA Cyber Fast Track
  - [www.cft.usma.edu](http://www.cft.usma.edu)



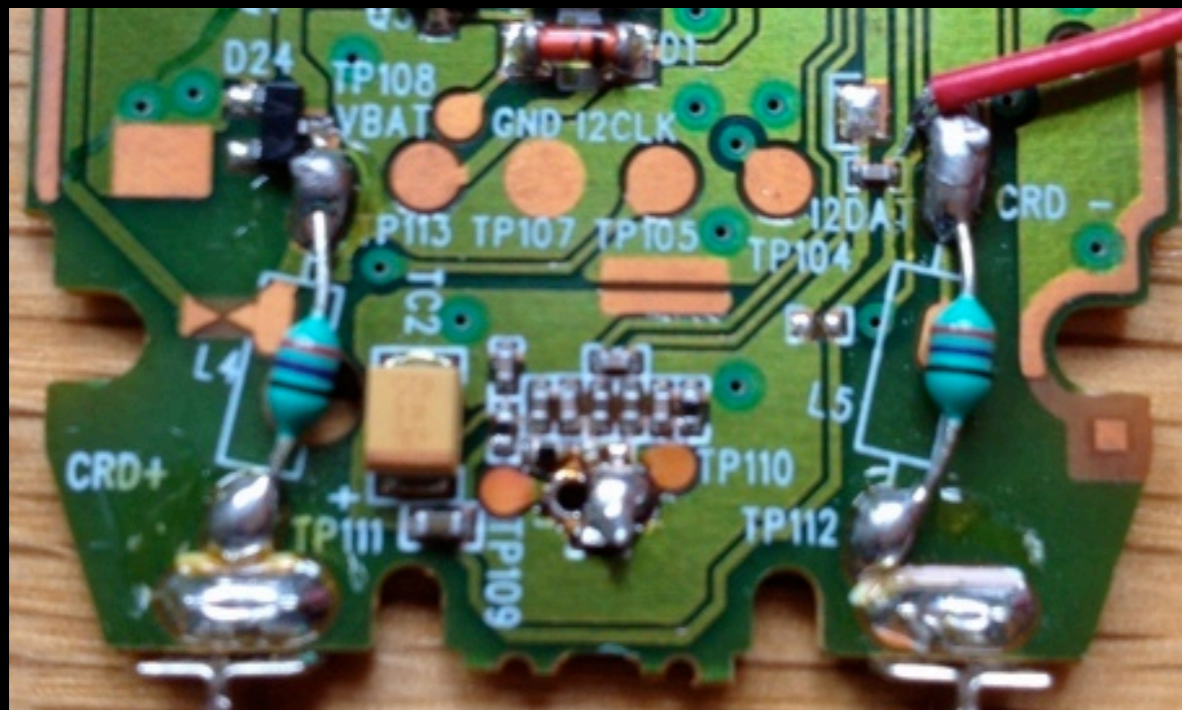
# Identifying Interfaces: External

- 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

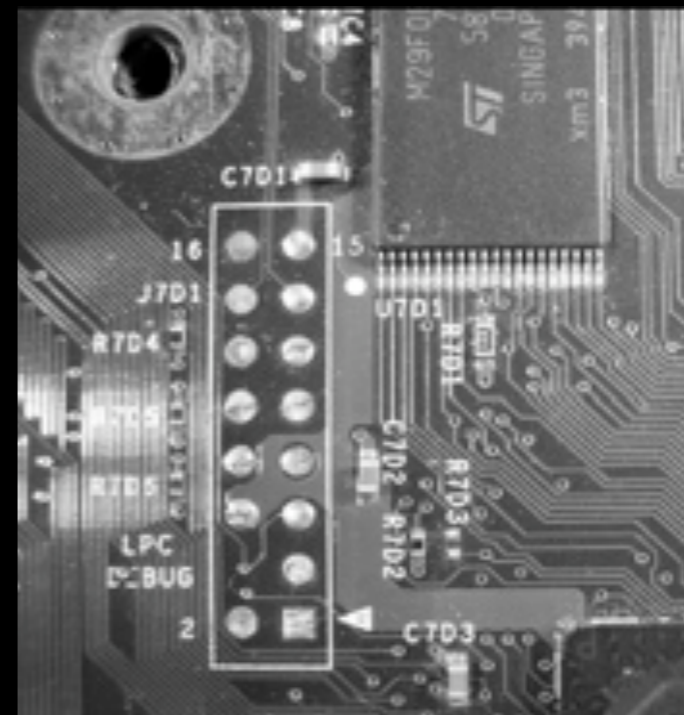
- Test points or unpopulated pads
- Silkscreen markings or notation
- Easy-to-access locations





# Identifying Interfaces: Internal 2

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



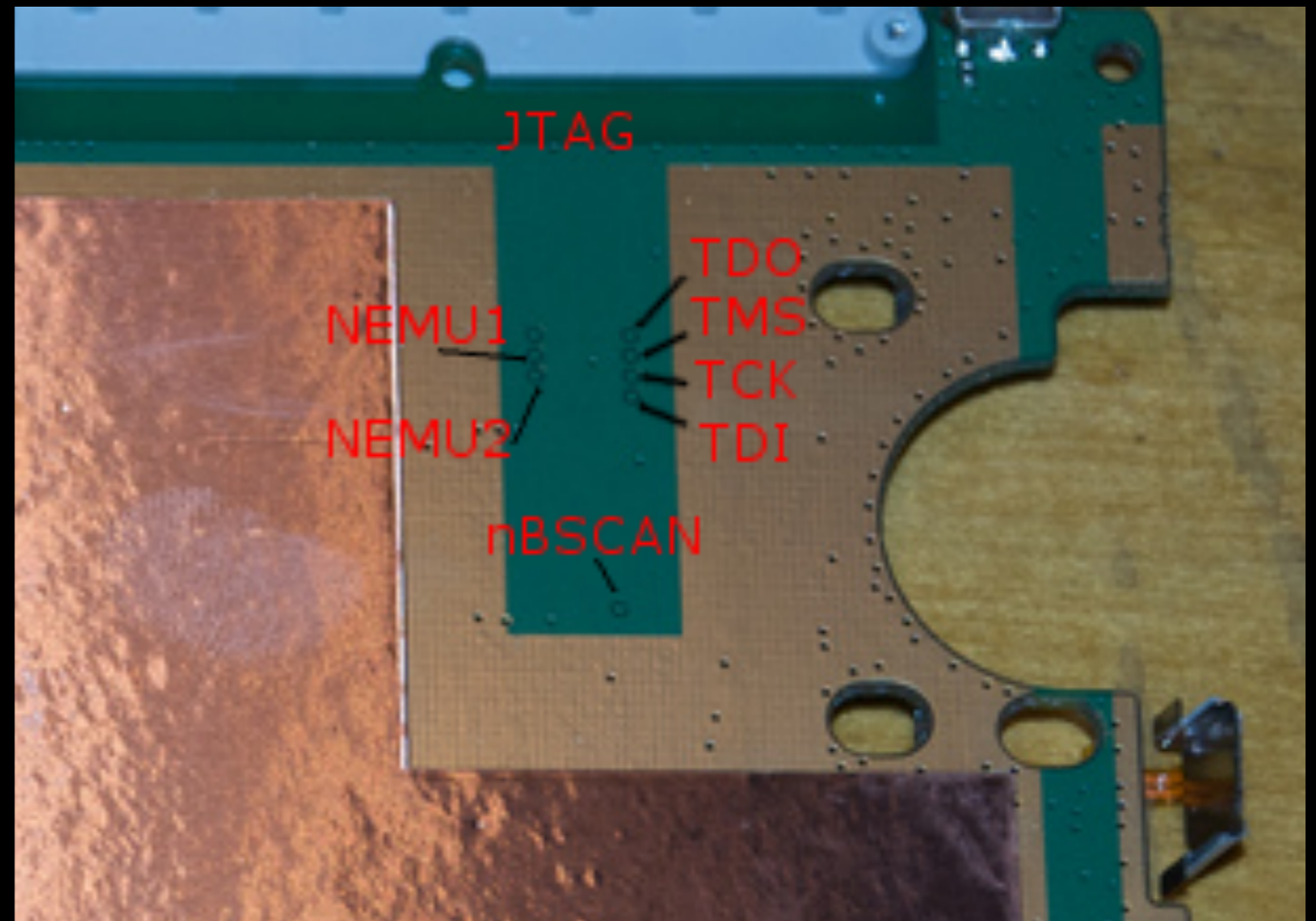
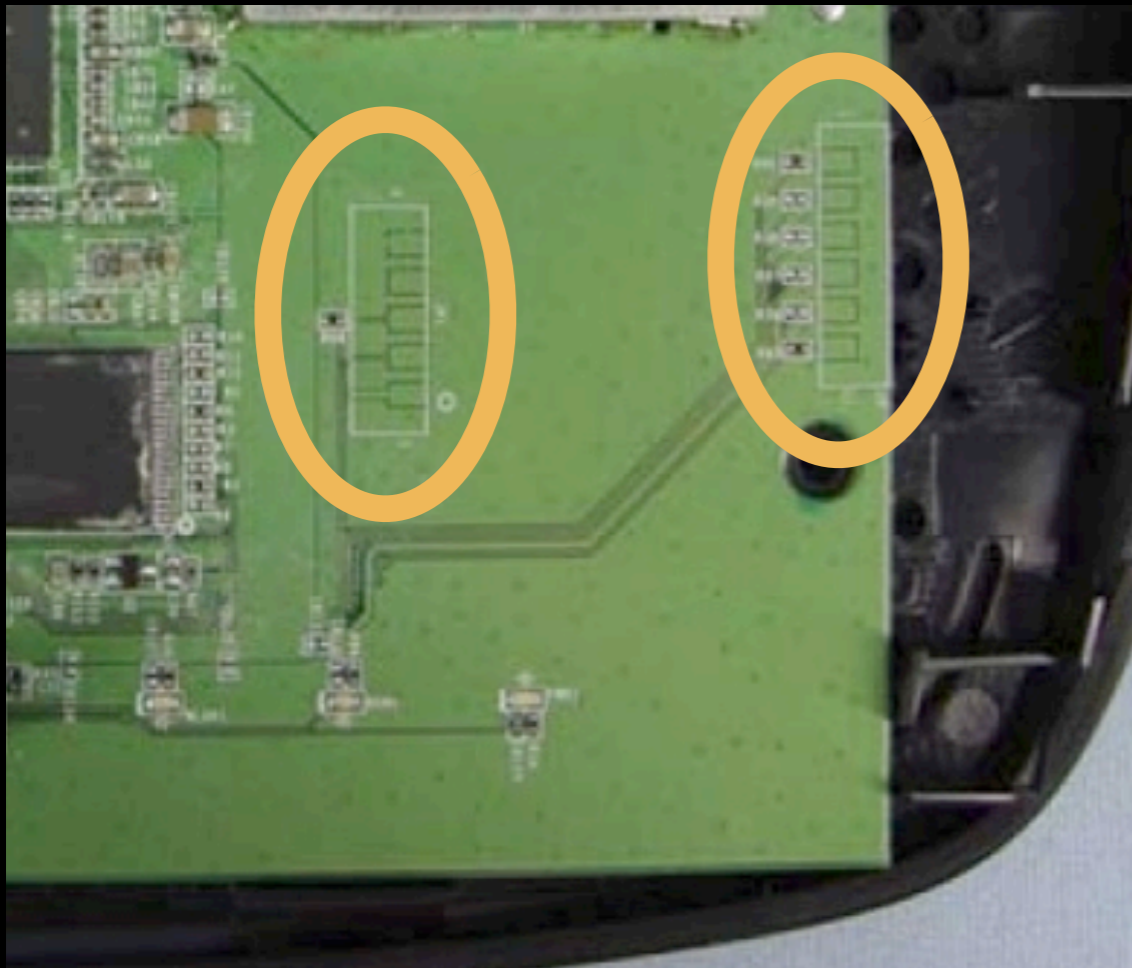
- ← [www.blackhat.com/html/bh-us-10/bh-us-10-archives.html#jack](http://www.blackhat.com/html/bh-us-10/bh-us-10-archives.html#jack)
- [www.nostarch.com/xboxfree](http://www.nostarch.com/xboxfree)





# Identifying Interfaces: Internal 4

- Might be covered by soldermask



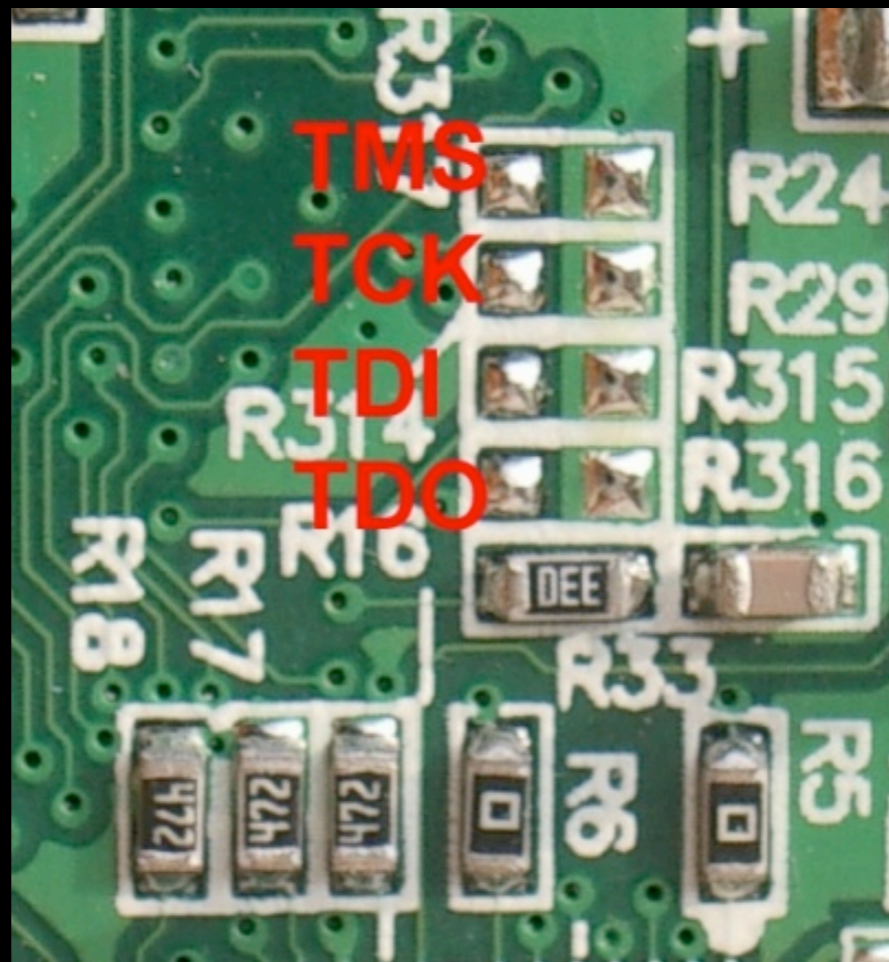
← Linksys WRT54G2 v1.3

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



# Identifying Interfaces: Internal 5

- More difficult to locate when available only on component pads



# Manually Determining Pin Function

- Identify test points/connector & target device
- Trace connections
  - Visually or w/ multimeter in continuity mode
  - For devices where pins aren't accessible (BGA), remove device or use X-ray
  - Use data sheet to match pin number to function
- Probe connections
  - Use oscilloscope or logic analyzer
  - Pull pins high or low, observe results, repeat
  - Logic state or number of pins can help to make educated guesses



# Design Requirements

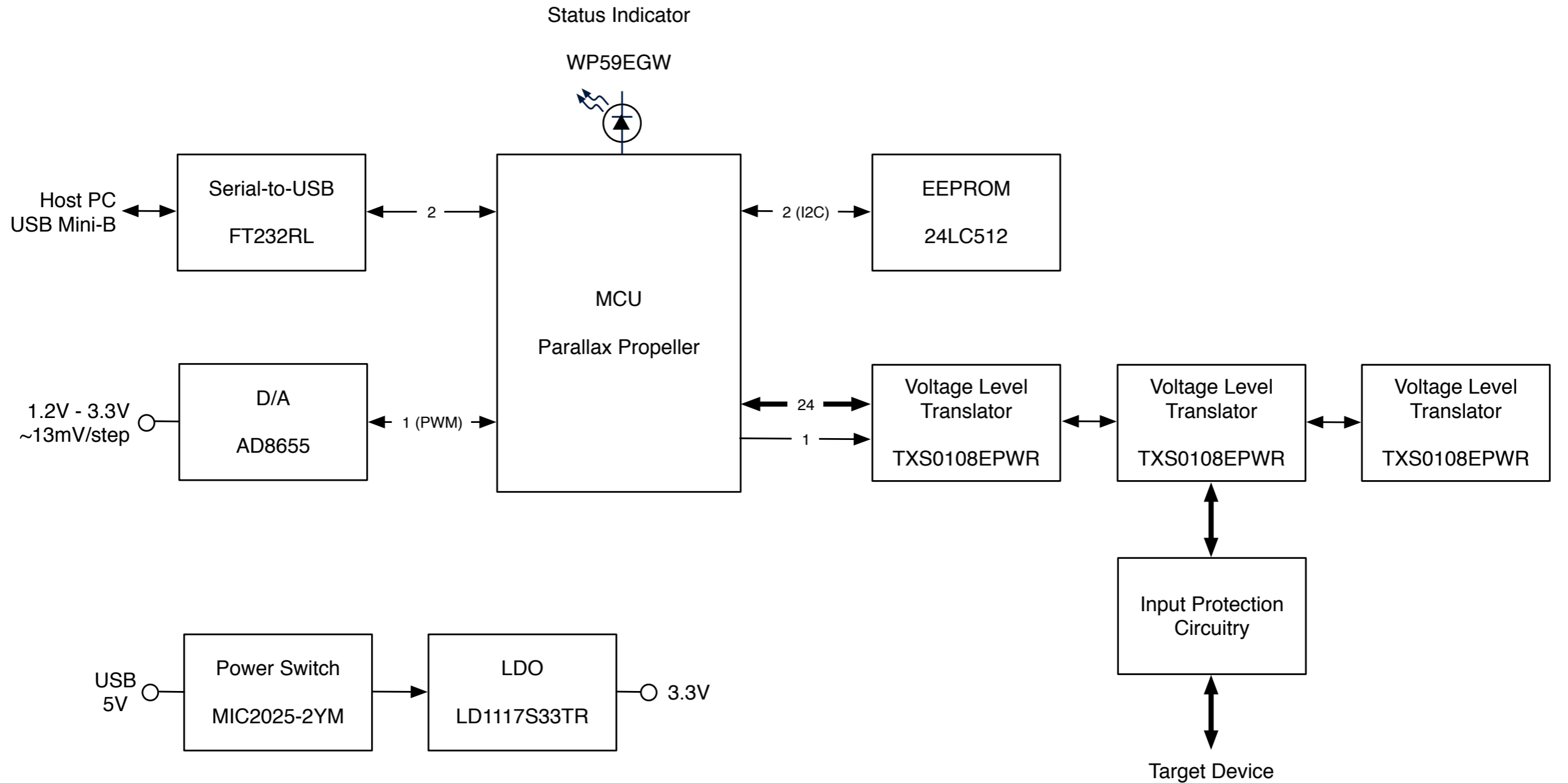
- Open source/hackable/expandable
- Simple command-based interface
- Input protection
- Adjustable target voltage
- Off-the-shelf components
- Hand solderable (if desired)



# Hardware



# Block Diagram





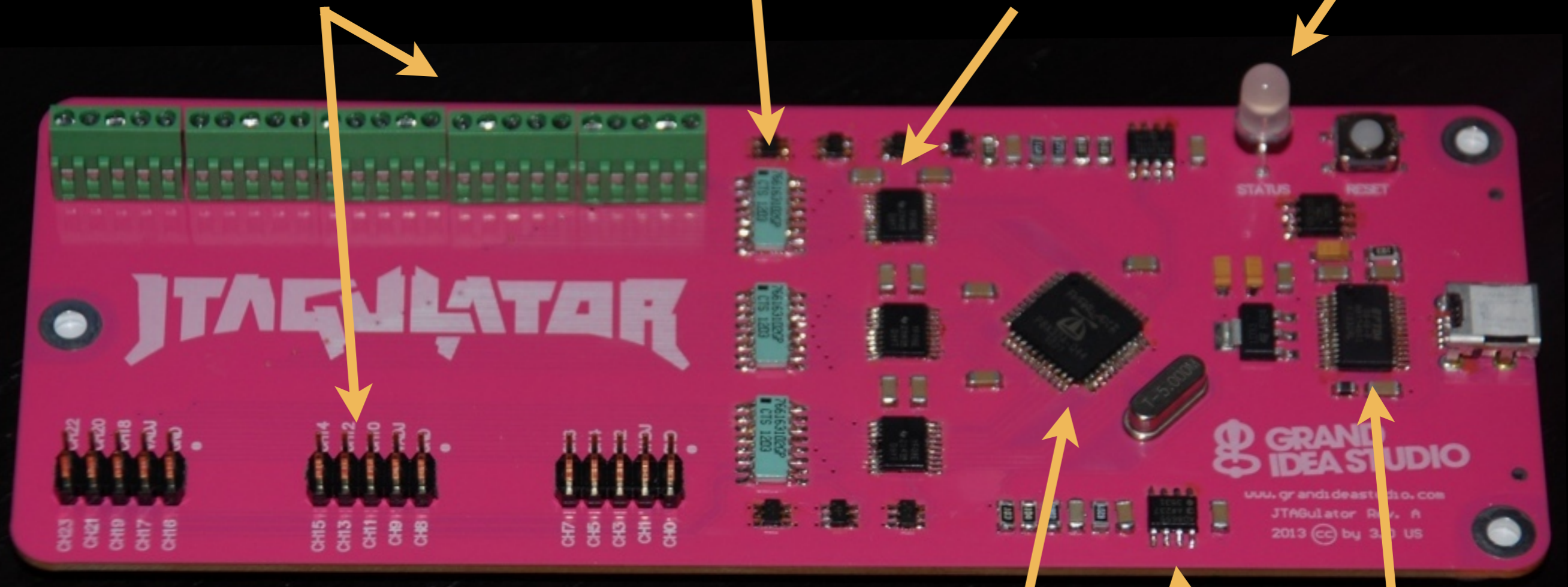
# PCB

Input protection

Target I/F (24 channels)

Level translation

Status



\*\*\* 2x5 headers compatible w/ Bus Pirate probes,  
[http://dangerousprototypes.com/docs/Bus\\_Pirate](http://dangerousprototypes.com/docs/Bus_Pirate)

Propeller

Op-Amp/DAC

USB



# Propeller/Core



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



\*\*\* INFORMATION: [www.parallax.com/propeller/](http://www.parallax.com/propeller/)

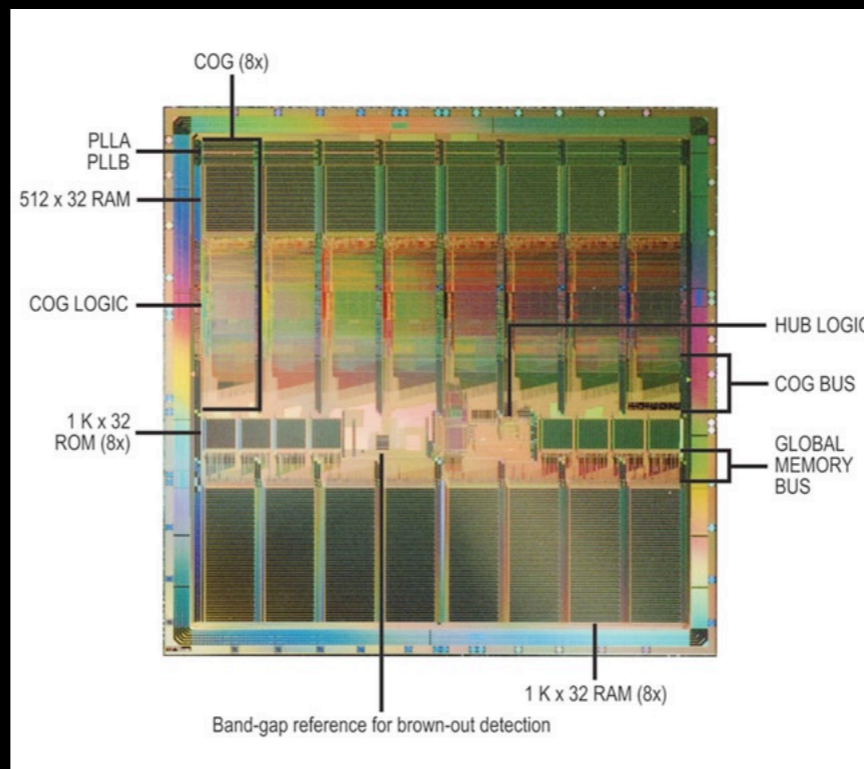
\*\*\* DISCUSSION FORUMS: <http://forums.parallax.com>

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



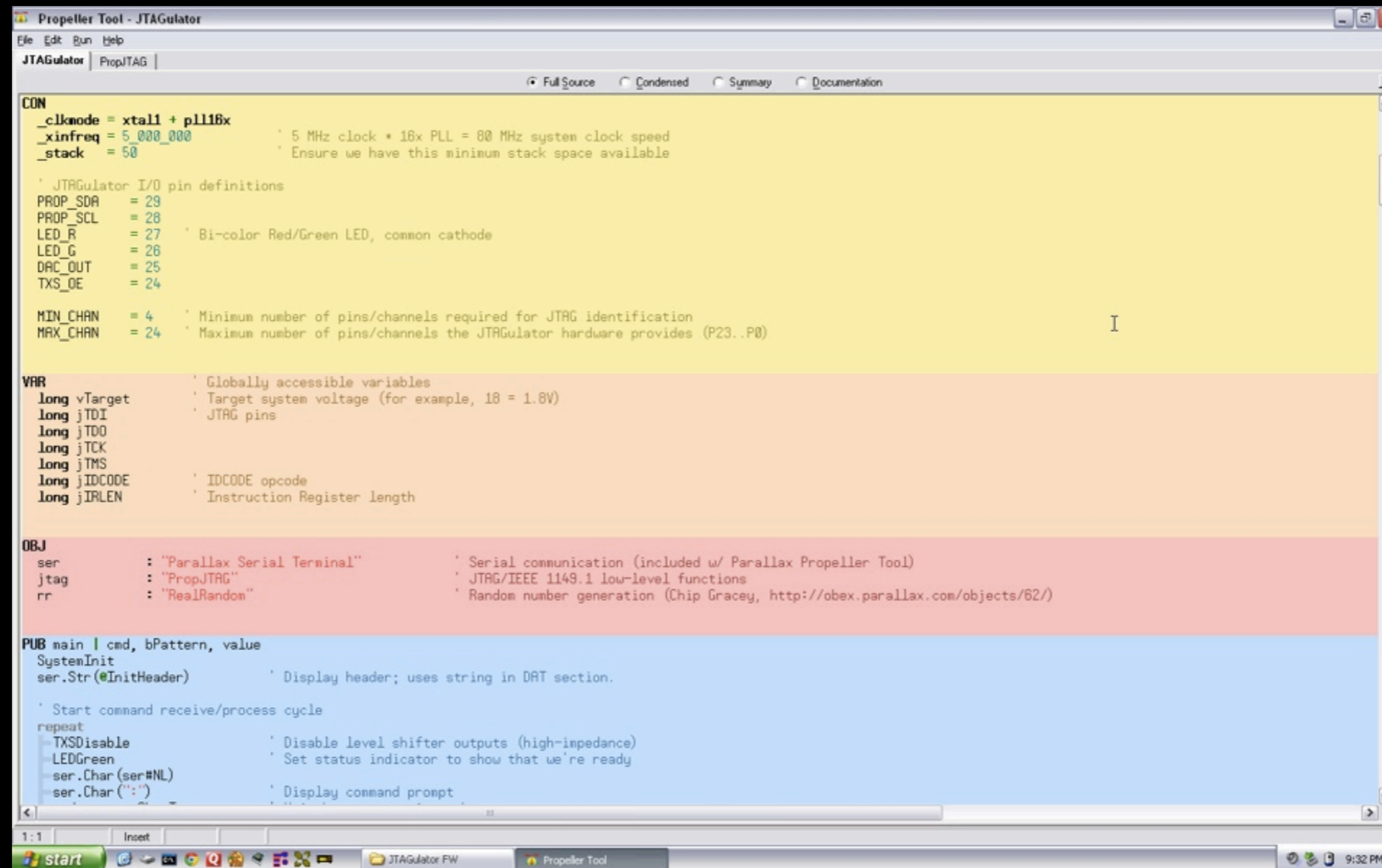
# Propeller/Core 2

- 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

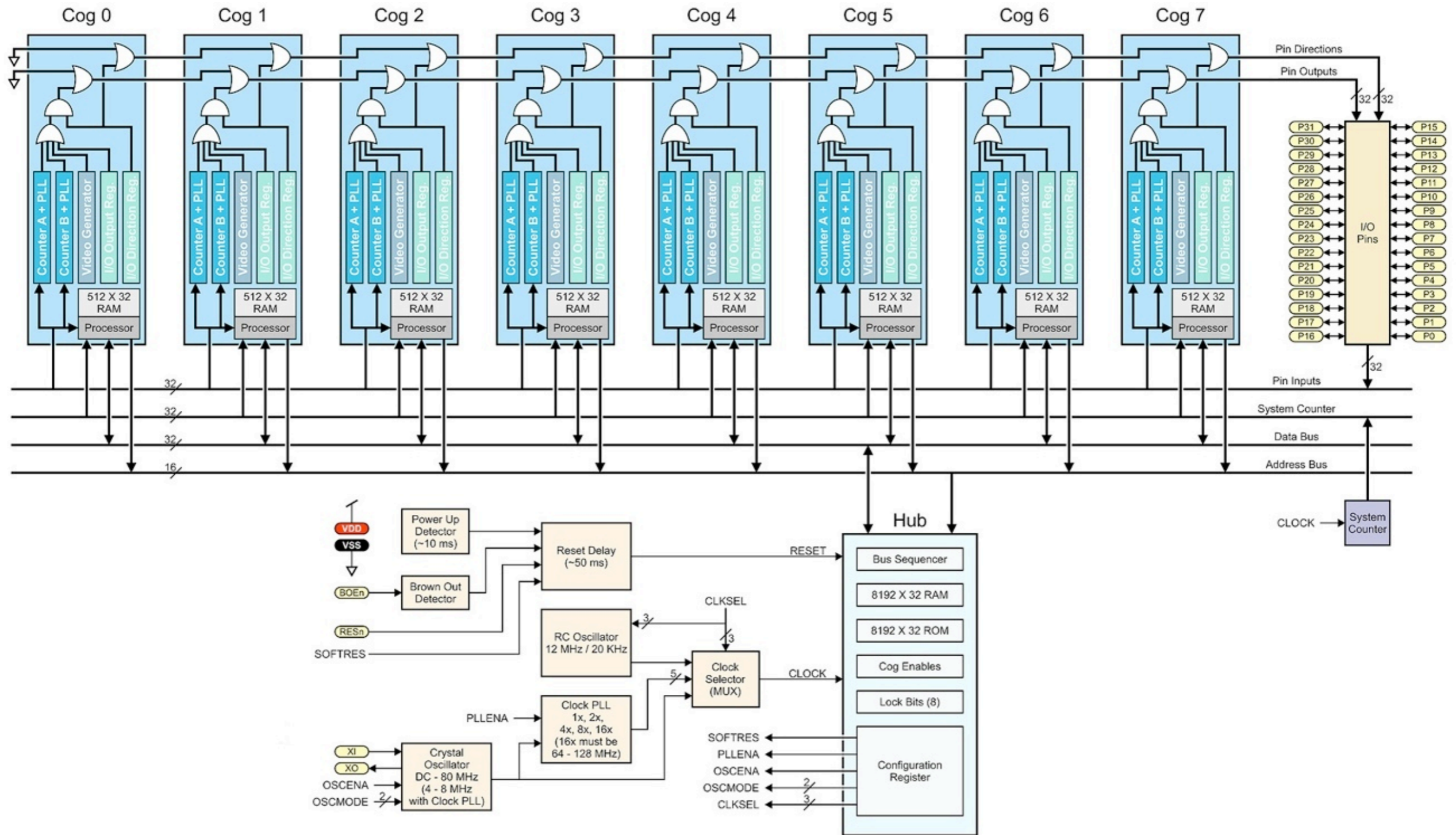
OBJ
ser : "Parallax Serial Terminal" * Serial communication (included w/ Parallax Propeller Tool)
jtag : "PropJTAG" * JTAG/IEEE 1149.1 low-level functions
rr : "RealRandom" * Random number generation (Chip Gracey, http://obex.parallax.com/objects/62/)

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

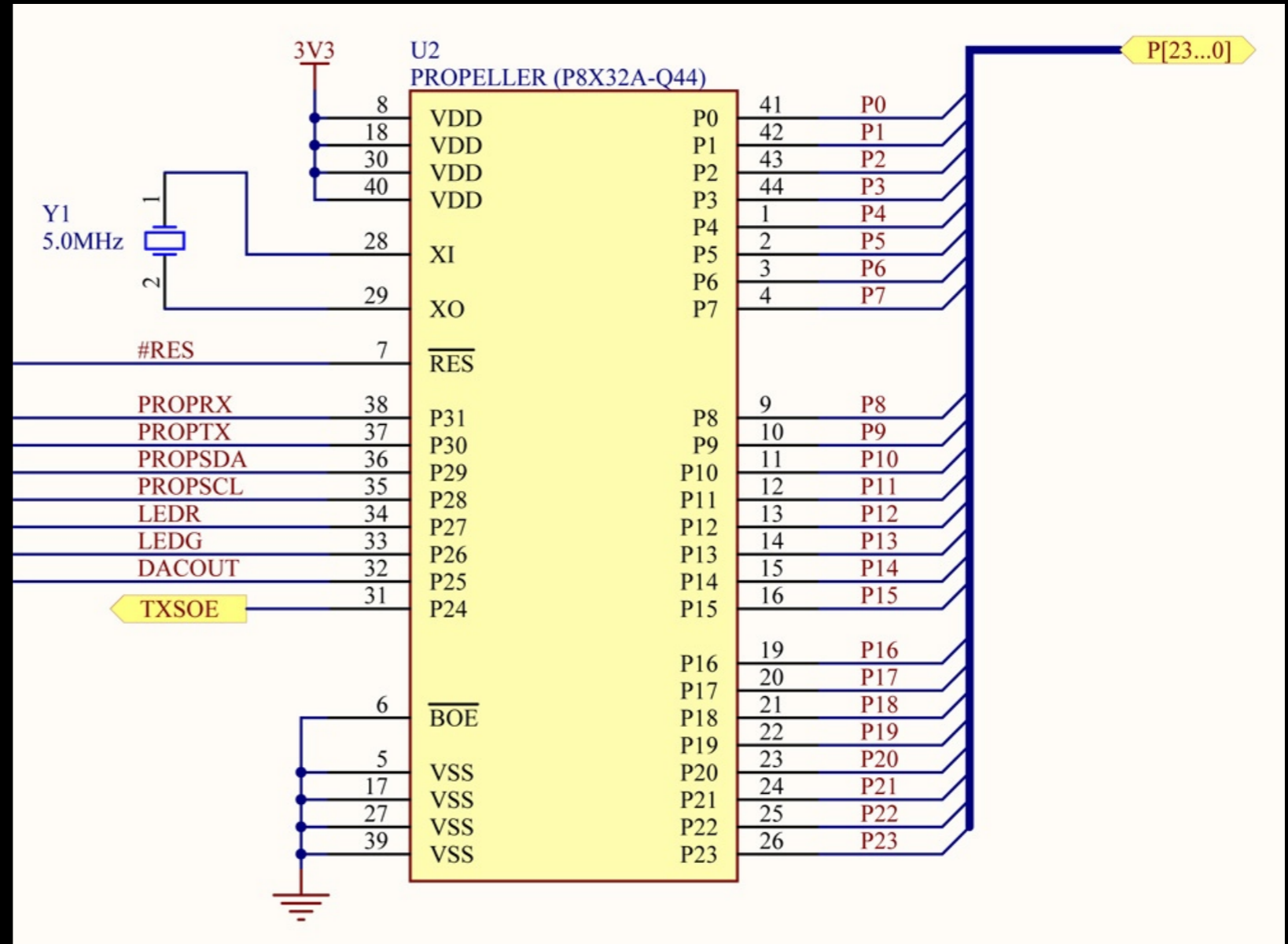
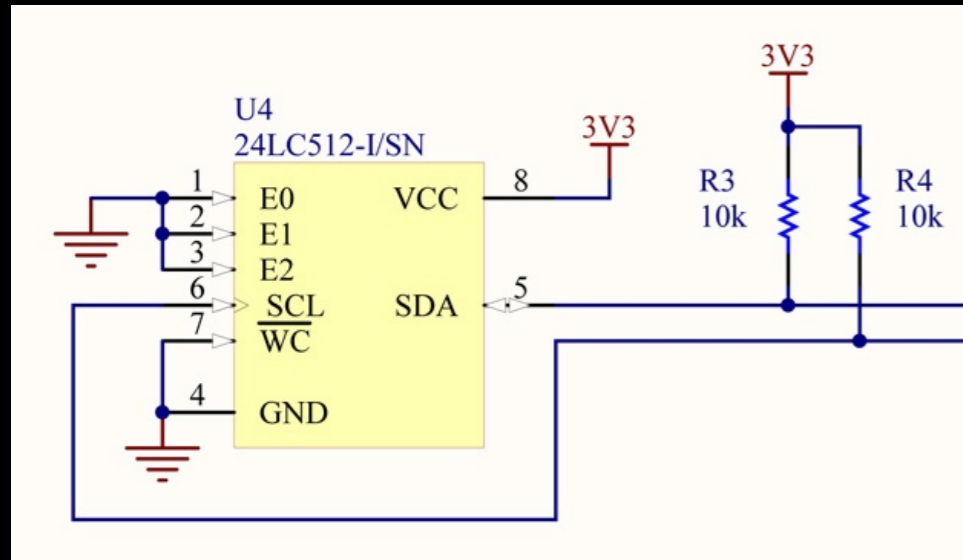
* Start command receive/process cycle
repeat
TXSDisable * Disable level shifter outputs (high-impedance)
LEDGreen * Set status indicator to show that we're ready
ser.Char(ser#NL)
ser.Char(":") * Display command prompt
```



# Propeller/Core 4



# Propeller/Core 5



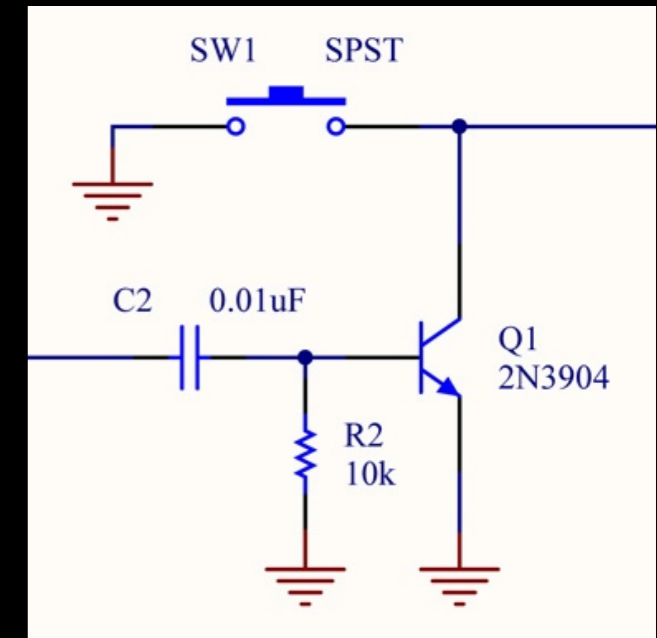
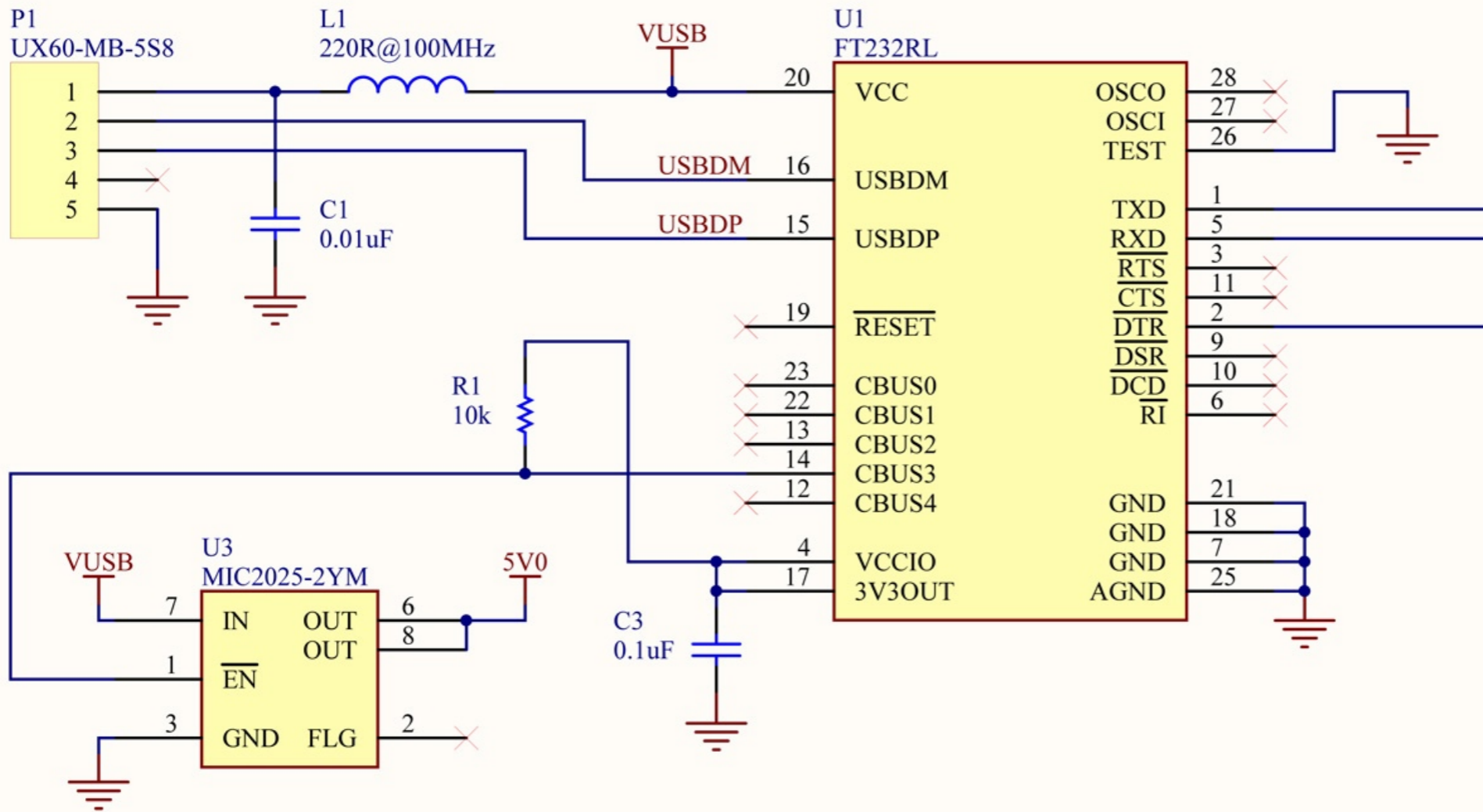
# USB Interface

- 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



# USB Interface 2

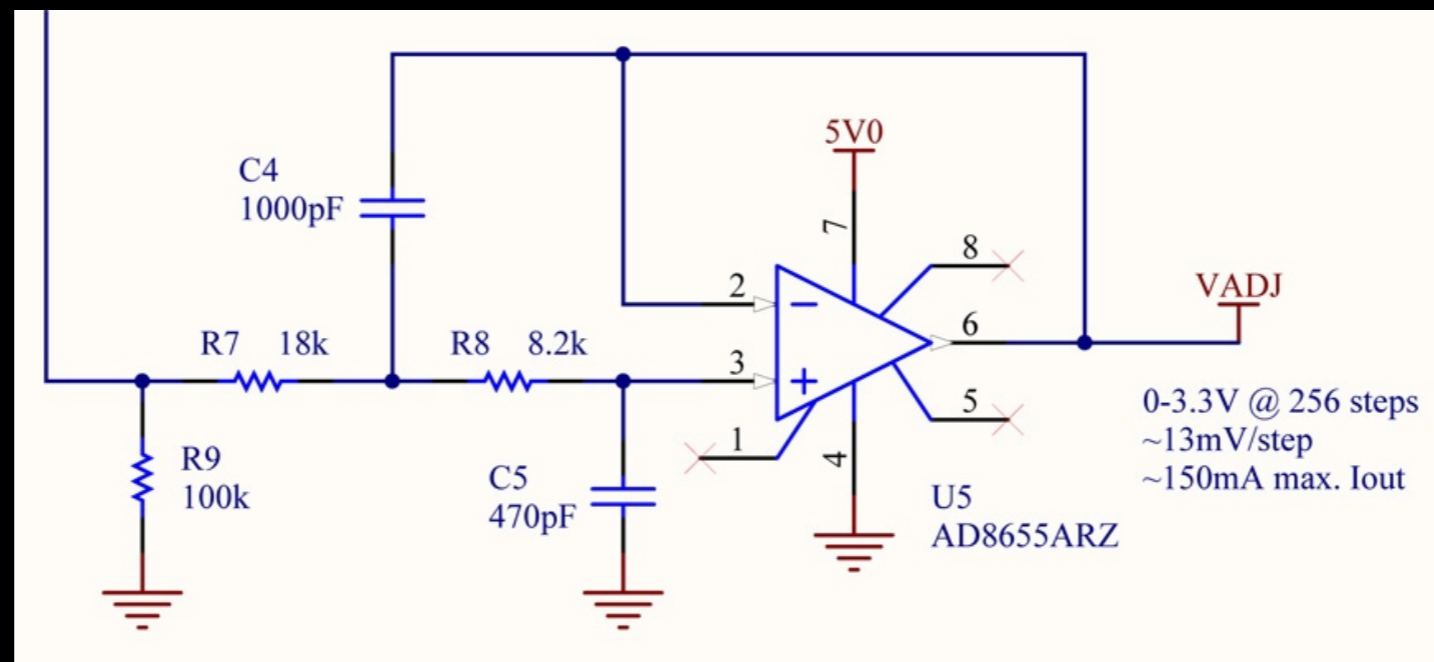
To Host  
USB Mini B





# 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 @  $V_o = 1.2V-3.3V$

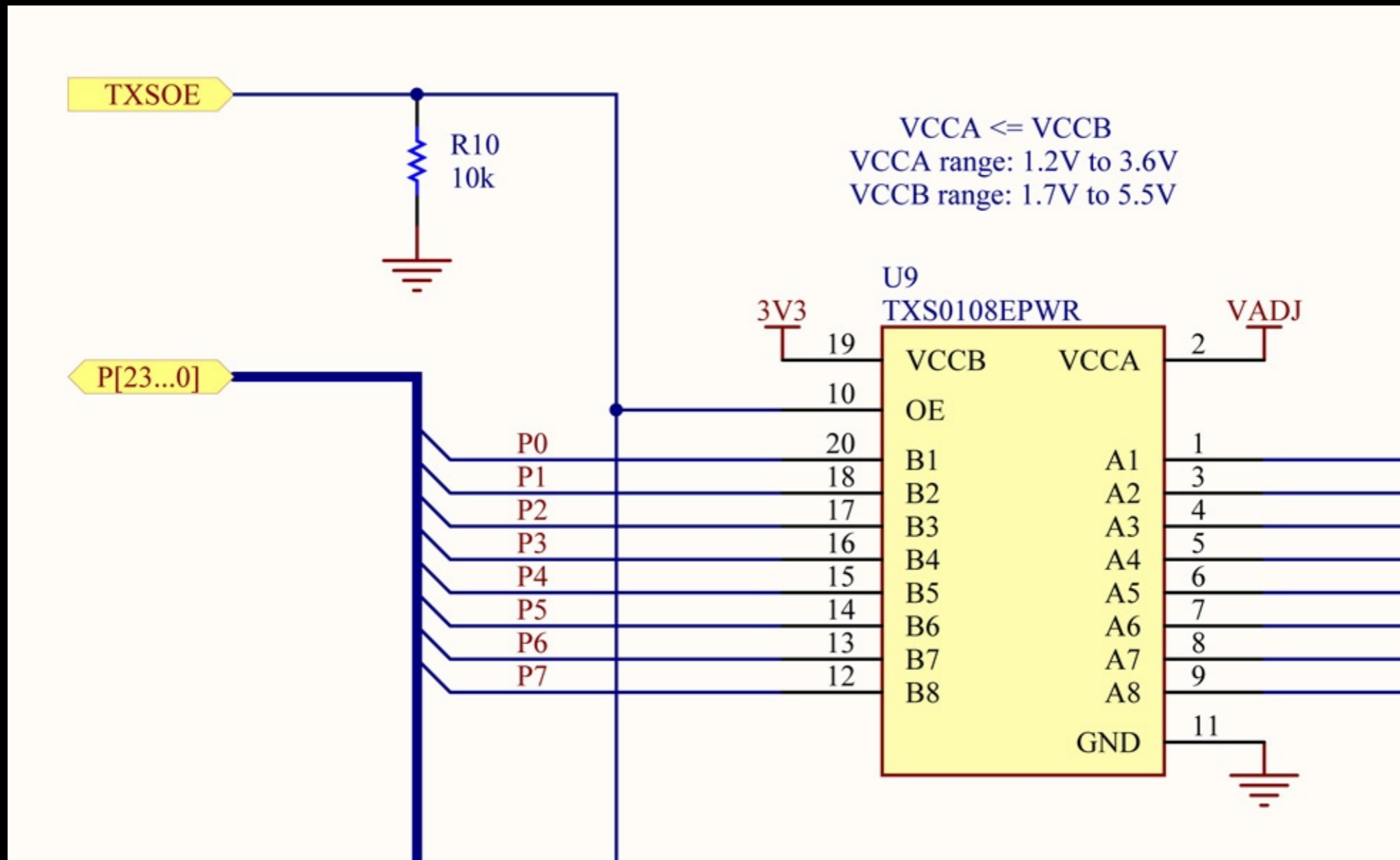


# Level Translation

- 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 $\Omega$  when driving low, 4k $\Omega$  when high)
  - Automatic signal direction detection
  - High-Z outputs when OE low  $\rightarrow$  will not interfere with target when not in use

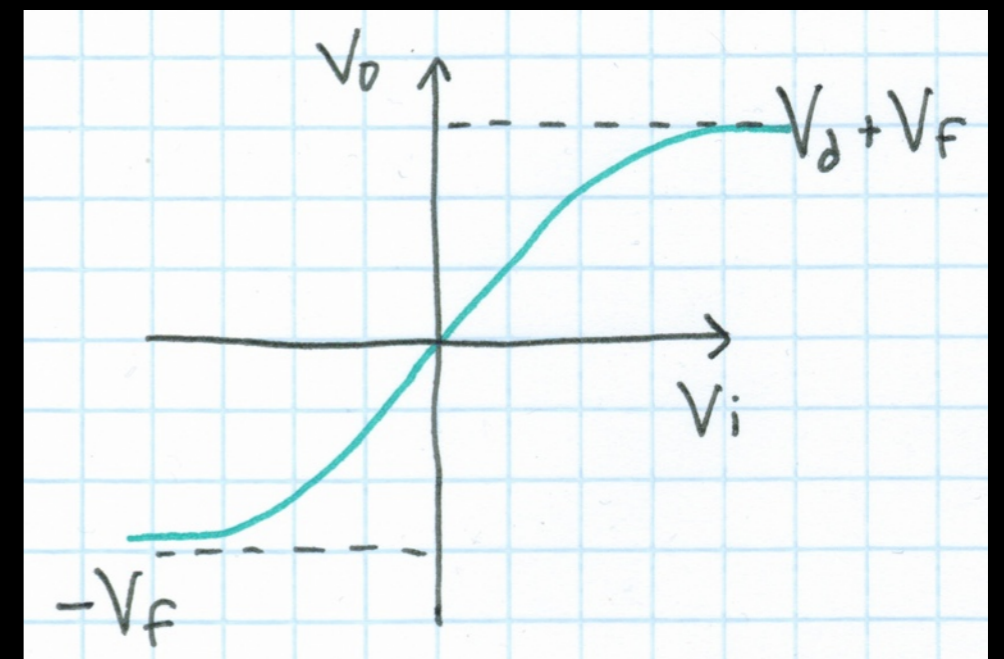
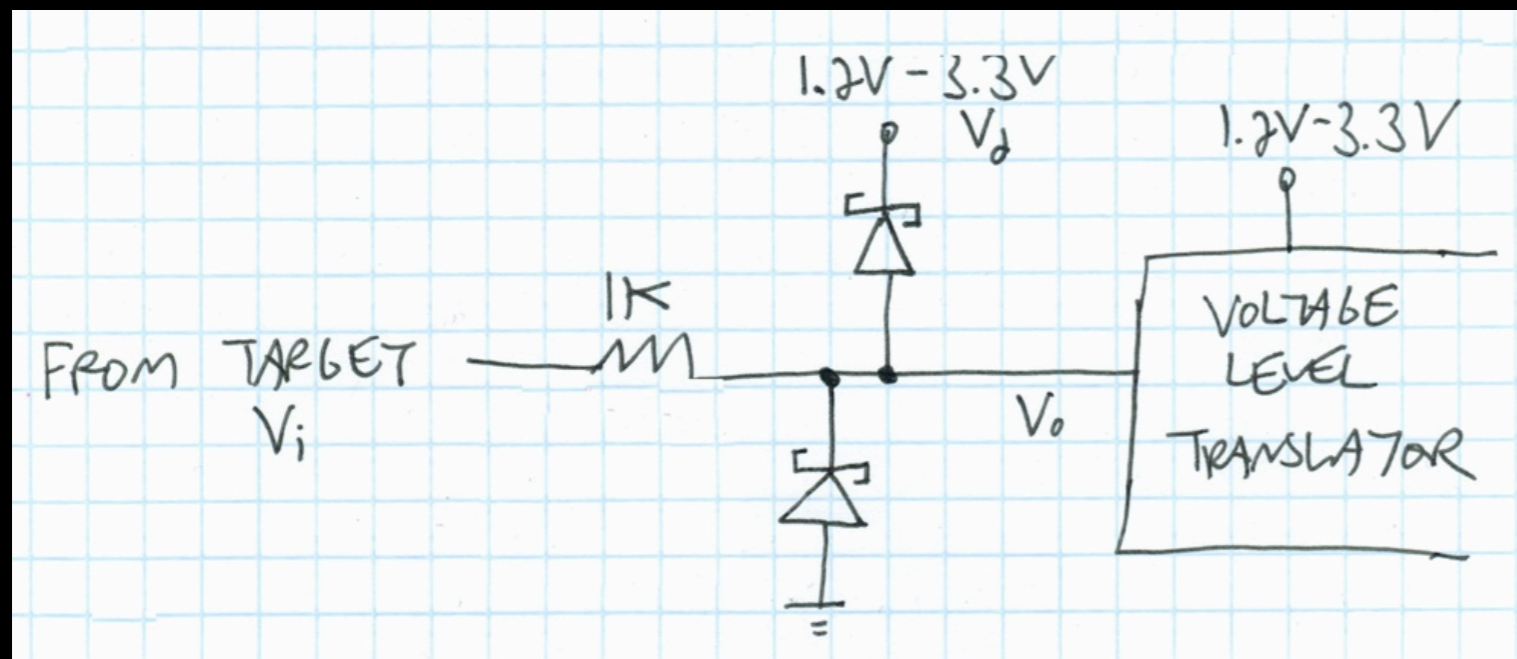


# Level Translation 2



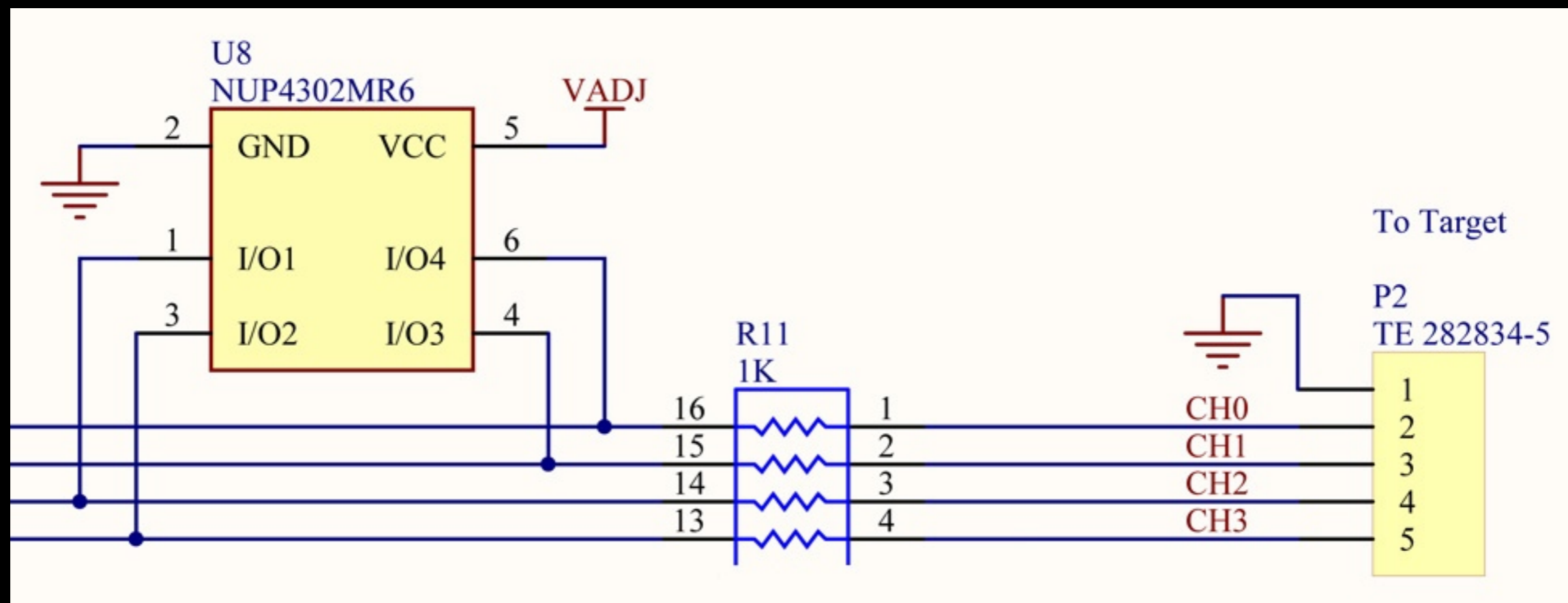
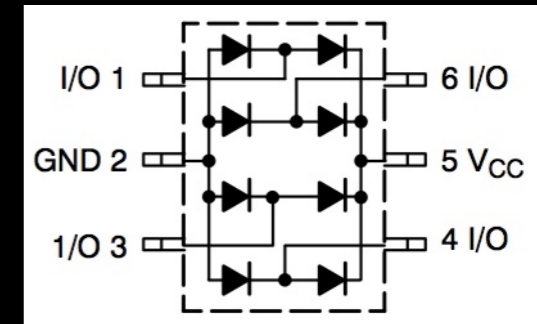
# Input Protection

- 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



# Input Protection 2

- NUP4302MR6 Schottky Diode Array
  - $V_f @ 1\text{mA} = 0.2\text{V typ.}, 0.35\text{V max.}$
  - $V_f @ 10\text{mA} = 0.25\text{V typ.}, 0.45\text{V max.}$
  - Alternate: SD103ASDM



# Bill-of-Materials

Item	Quantity	Reference	Manufacturer	Manuf. Part #	Distributor	Distrib. Part #	Description
1	2	C1, C2	Kemet	C1206C103K5RACTU	Digi-Key	399-1234-1-ND	Capacitor, 0.01uF ceramic, 10%, 50V, X7R, 1206
2	14	C3, C6, C9, C11, C12, C13, C14, C15, C17, C18, C19, C20, C21, C22	Kemet	C1206C104K5RACTU	Digi-Key	399-1249-1-ND	Capacitor, 0.1uF ceramic, 10%, 50V, X7R, 1206
3	1	C4	Yageo	CC1206KRX7R9BB102	Digi-Key	311-1170-1-ND	Capacitor, 1000pF ceramic, 10%, 50V, X7R, 1206
4	1	C5	Yageo	CC1206KRX7R9BB471	Digi-Key	311-1167-1-ND	Capacitor, 470pF ceramic, 10%, 50V, X7R, 1206
5	1	C7	Kemet	T491A106M016AS	Digi-Key	399-3687-1-ND	Capacitor, 10uF tantalum, 20%, 16V, size A
6	2	C8, C10	Kemet	T491A475K016AT	Digi-Key	399-3697-1-ND	Capacitor, 4.7uF tantalum, 10%, 16V, size A
7	1	D1	Kingbright	WP59EGW	Digi-Key	754-1232-ND	LED, Red/Green Bi-Color, T-1 3/4 (5mm)
8	1	L1	TDK	MPZ2012S221A	Digi-Key	445-1568-1-ND	Inductor, Ferrite Bead, 220R@100MHz, 3A, 0805
9	1	P1	Hirose Electric	UX60-MB-5S8	Digi-Key	H2960CT-ND	Connector, Mini-USB, 5-pin, SMT w/ PCB mount
10	5	P2, P3, P4, P5, P6	TE Connectivity	282834-5	Digi-Key	A98336-ND	Connector, Terminal Block, 5-pin, side entry, 0.1" P
11	3	P7, P8, P9	3M	961210-6404-AR	Digi-Key	3M9460-ND	Header, Dual row, Vertical header, 2x5-pin, 0.1" P
12	1	Q1	Fairchild	MMBT3904	Digi-Key	MMBT3904FSCT-ND	Transistor, NPN, 40V, 200mA, SOT23-3
13	5	R1, R2, R3, R4, R10	Any	Any	Digi-Key	P10KECT-ND	Resistor, 10k, 5%, 1/4W, 1206
14	1	R5	Any	Any	Digi-Key	P470ECT-ND	Resistor, 470 ohm, 5%, 1/4W, 1206
15	1	R6	Any	Any	Digi-Key	P270ECT-ND	Resistor, 270 ohm, 5%, 1/4W, 1206
16	1	R7	Any	Any	Digi-Key	P18.0KFCT-ND	Resistor, 18k, 1%, 1/4W, 1206
17	1	R8	Any	Any	Digi-Key	P8.20KFCT-ND	Resistor, 8.2k, 1%, 1/4W, 1206
18	1	R9	Any	Any	Digi-Key	P100KECT-ND	Resistor, 100k, 5%, 1/4W, 1206
19	3	R11, R12, R13	Bourns	4816P-1-102LF	Digi-Key	4816P-1-102LFCT-ND	Resistor, Array, 8 isolated, 1k, 2%, 1/6W, SOIC16
20	1	SW1	C&K	KSC201JLFS	Digi-Key	401-1756-1-ND	Switch, SPST, Momentary, 120gf, 6.2 x 6.2mm, J-Lead
21	1	U1	FTDI	FT232RL-REEL	Digi-Key	768-1007-1-ND	IC, USB-to-UART Bridge, SSOP28
22	1	U2	Parallax	P8X32A-Q44	Digi-Key	P8X32A-Q44-ND	IC, Microcontroller, Propeller, LQFP44
23	1	U3	Micrel	MIC2025-2YM	Digi-Key	576-1058-ND	IC, Power Distribution Switch, Single-channel, SOIC8
24	1	U4	Microchip	24LC512-I/SN	Digi-Key	24LC512-I/SN-ND	IC, Memory, Serial EEPROM, 64KB, SOIC8
25	1	U5	Analog Devices	AD8655ARZ	Digi-Key	AD8655ARZ-ND	IC, Op. Amp., CMOS, Rail-to-rail, 220mA Iout, SOIC8
26	1	U6	ST Microelectronics	LD1117S33CTR	Digi-Key	497-1241-1-ND	IC, Voltage Regulator, LDO, 3.3V@800mA, SOT223
27	6	U7, U8, U10, U11, U13, U14	ON Semiconductor	NUP4302MR6T1G	Digi-Key	NUP4302MR6T1GOSCT-ND	IC, Schottky Diode Array, 4 channel, TSOP6
28	3	U9, U12, U15	Texas Instruments	TXS0108EPWR	Digi-Key	296-23011-1-ND	IC, Level Translator, Bi-directional, TSSOP20
29	1	Y1	ECS	ECS-50-18-4XEN	Digi-Key	XC1738-ND	Crystal, 5.0MHz, 18pF, HC49/US
30	1	PCB	Any	JTAG B	N/A	N/A	PCB, Fabrication

- All components from Digi-Key
- Total cost per unit = \$50.73



# Firmware



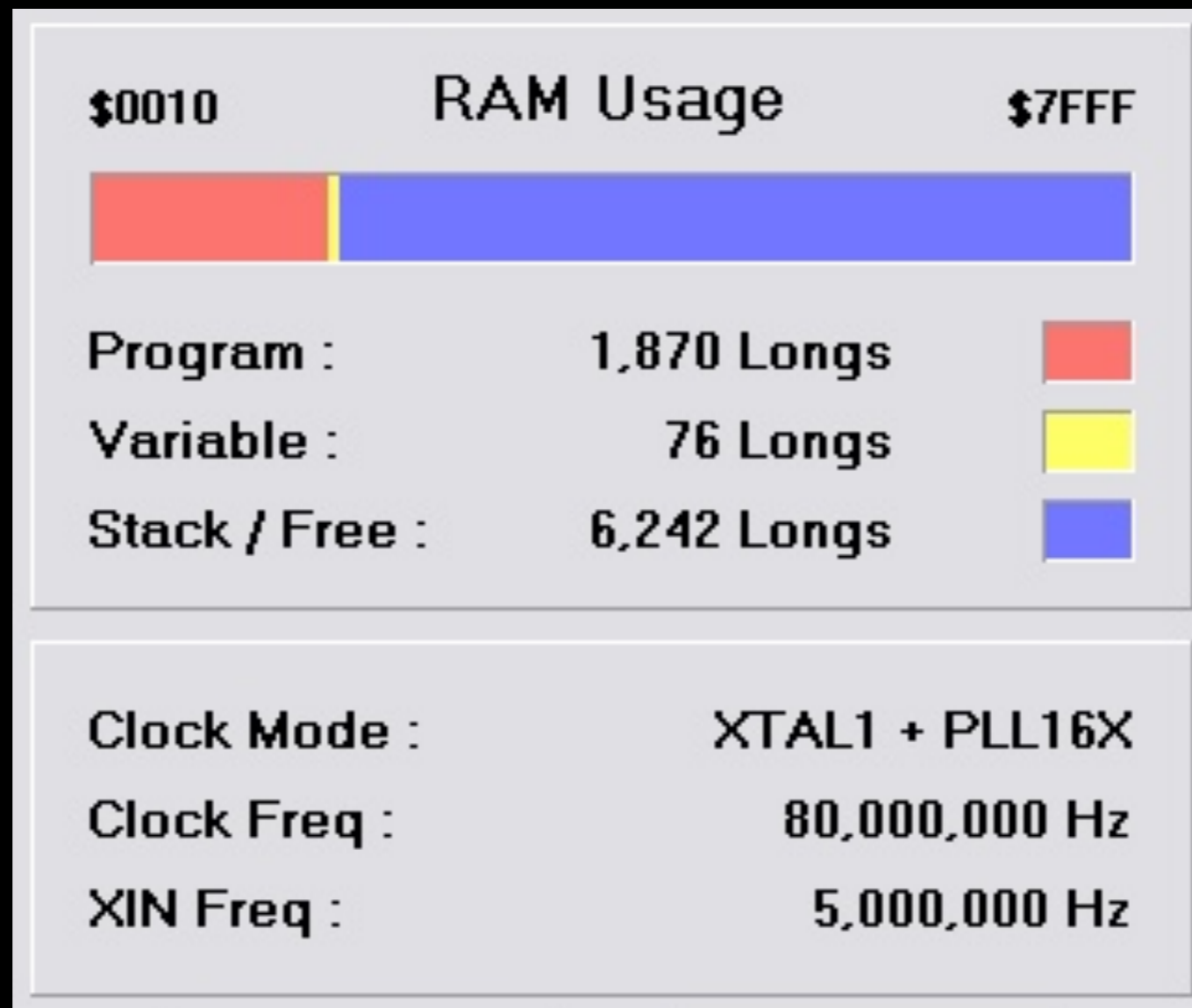
# Source Tree

```
JTAGulator.spin
├── Parallax Serial Terminal.spin
├── RealRandom.spin
├── PropJTAG.spin
└── JDCogSerial.spin
```





# Propeller Resources



# 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](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 2

- Multiple devices can be "chained" together for communication to all via a single JTAG port
  - Even multiple dies within the same chip package
  - Different vendors may not play well together
- Development environments abstract low-level functionality from the user
  - Implementations are device- or family-specific
  - As long as we can locate the interface/pinout, let other tools do the rest



# JTAG 3



\*\*\* [ruxconbreakpoint.com/assets/slides/pres\\_sprite\\_tm.pdf](http://ruxconbreakpoint.com/assets/slides/pres_sprite_tm.pdf)

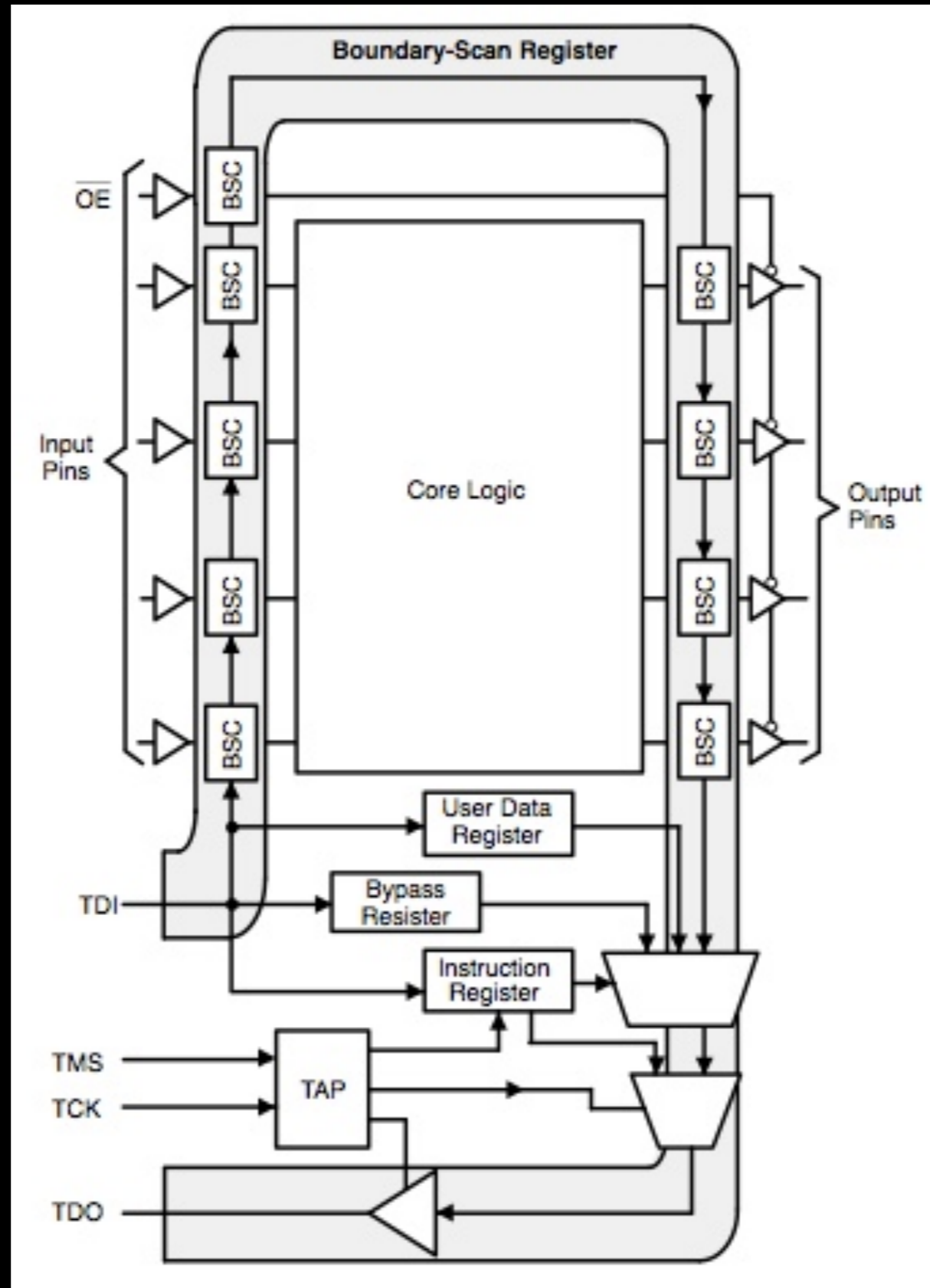


# JTAG: Architecture

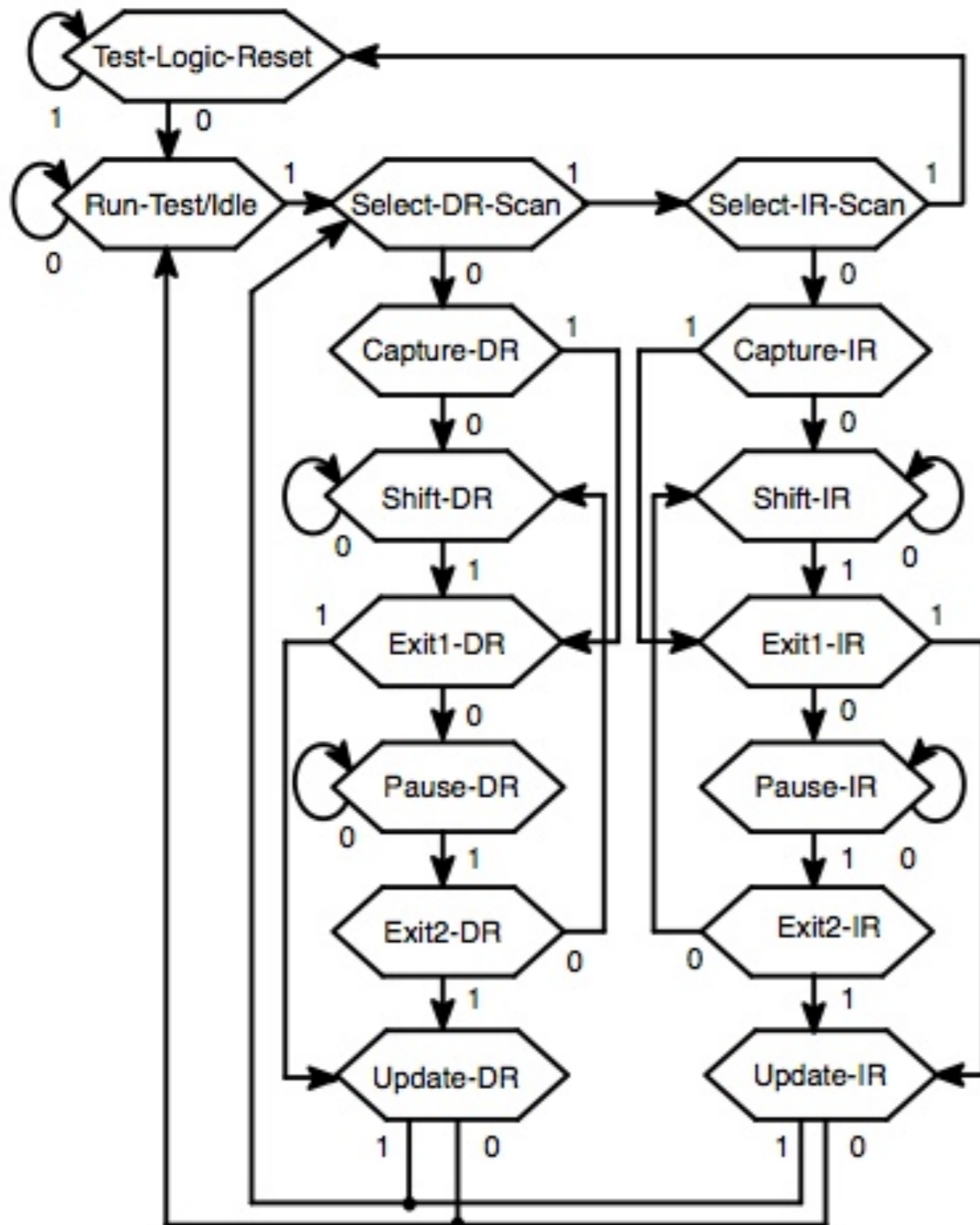
- Synchronous serial interface
  - TDI = Data In (to target device)
  - ← TDO = Data Out (from target device)
  - TMS = Test Mode Select
  - TCK = Test Clock
  - /TRST = Test Reset (optional for async reset)
- Test Access Port (TAP) w/ Shift Registers
  - Instruction ( $\geq 2$  bit wide)
  - Data
 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



# JTAG: Architecture 2



# 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: Instructions

Name	Required?	Opcode	Description
BYPASS	Y	All 1s	Bypass on-chip system logic. Allows serial data to be transferred from TDI to TDO without affecting operation of the IC.
SAMPRE	Y	Varies	Used for controlling (preload) or observing (sample) the signals at device pins. Enables the boundary scan register.
EXTEST	Y	All 0s	Places the IC in external boundary test mode. Used to test device interconnections. Enables the boundary scan register.
INTEST	N	Varies	Used for static testing of internal device logic in a single-step mode. Enables the boundary scan register.
RUNBIST	N	Varies	Places the IC in a self-test mode and selects a user-specified data register to be enabled.
CLAMP	N	Varies	Sets the IC outputs to logic levels as defined in the boundary scan register. Enables the bypass register.
HIGHZ	N	Varies	Sets all IC outputs to a disabled (high impedance) state. Enables the bypass register.
IDCODE	N	Varies	Enables the 32-bit device identification register. Does not affect operation of the IC.
USERCODE	N	Varies	Places user-defined information into the 32-bit device identification register. Does not affect operation of the IC.



# 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](http://www.jtagbox.com)
- **H-JTAG**
  - [www.hjtag.com/en/](http://www.hjtag.com/en/)
- **Flyswatter2**
  - [www.tincantools.com](http://www.tincantools.com)
- **Bus Blaster (open source)**
  - [http://dangerousprototypes.com/docs/Bus\\_Blaster](http://dangerousprototypes.com/docs/Bus_Blaster)
- **Wiggler or compatible (parallel port)**
  - [ftp://www.keith-koep.com/pub/arm-tools/jtag/jtag05\\_sch.pdf](ftp://www.keith-koep.com/pub/arm-tools/jtag/jtag05_sch.pdf)



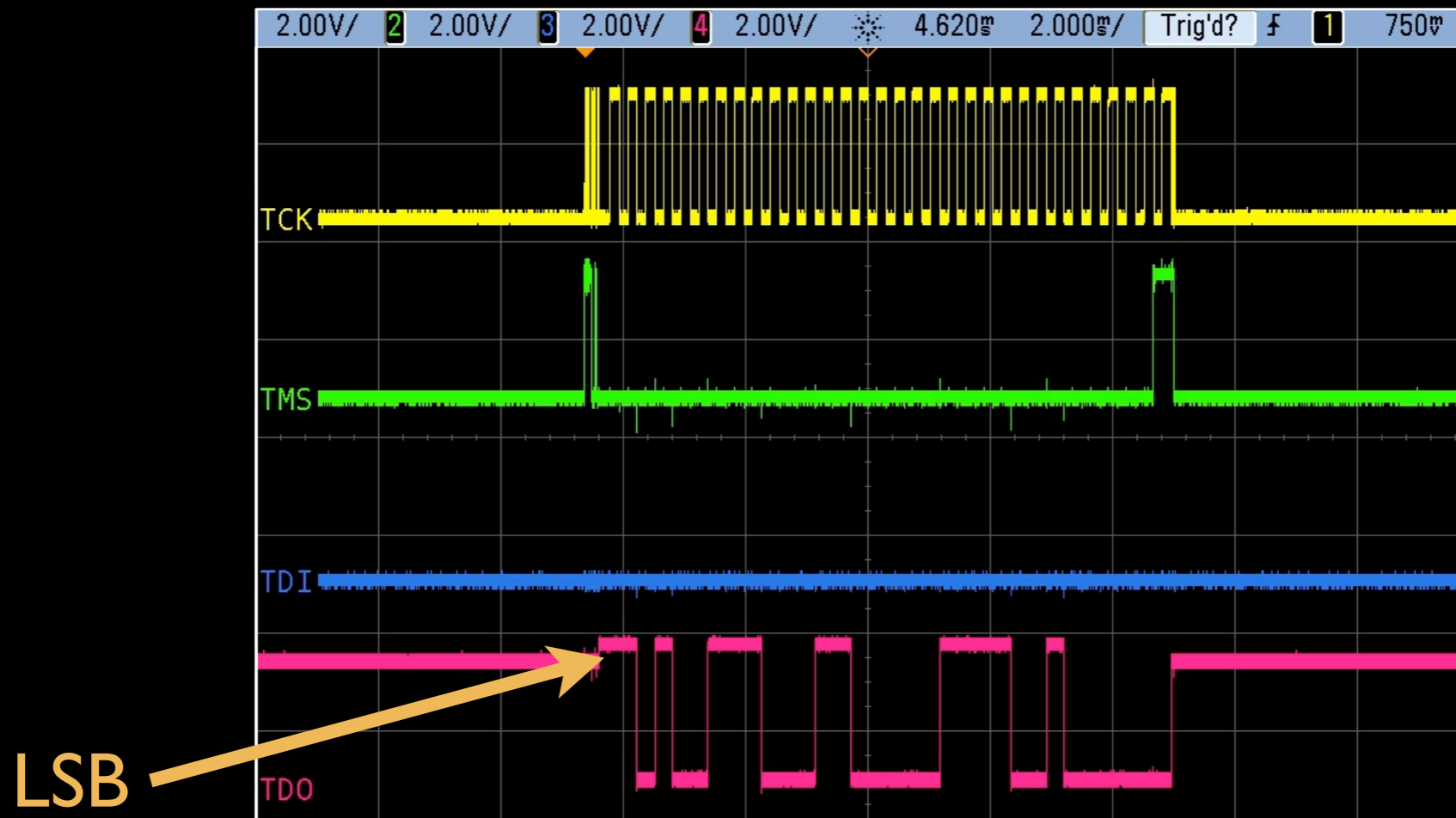
# JTAG: SW Tools

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



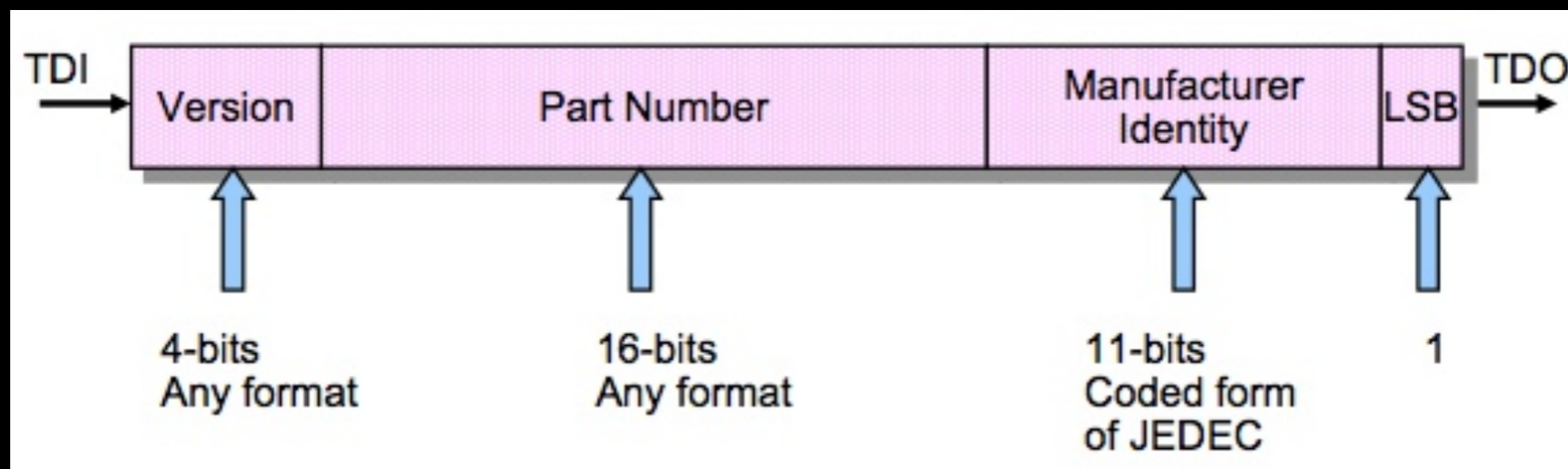
# IDCODE Scan

- 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



# IDCODE Scan 2

- Device ID values vary with part/family/vendor
  - Locate in data sheets, BSDL files, reference code, etc.
- Manufacturer ID provided by JEDEC
  - Each manufacturer assigned a unique identifier
  - Can use to help validate that proper IDCODE was retrieved
  - <http://www.jedec.org/standards-documents/results/jep106>



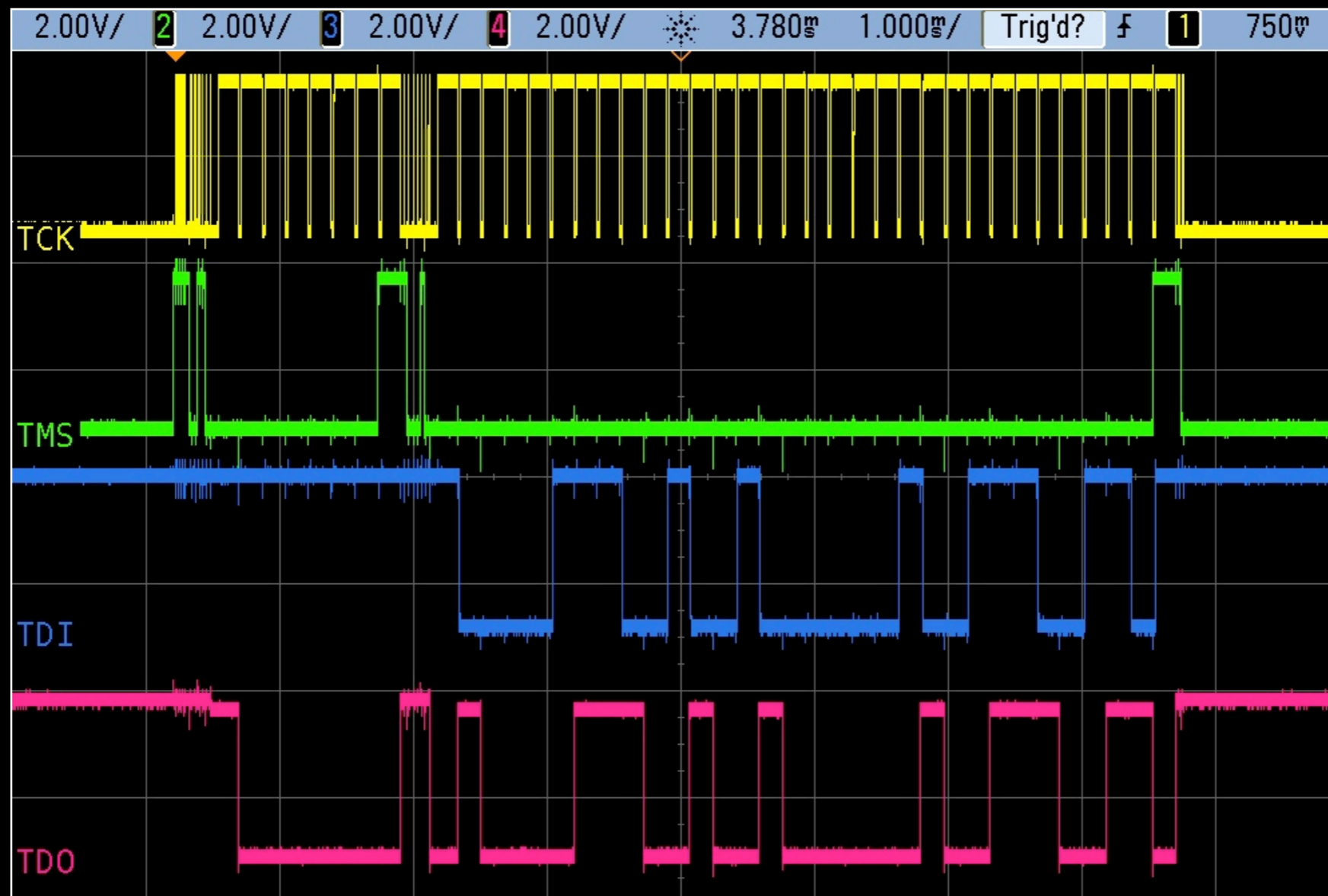
# IDCODE Scan 3

- Ask user for number of channels to use
- For every possible pin permutation (except TDI)
  - Set unused channels to output high (in case of any active low reset pins)
  - Configure JTAG pins to use on the Propeller
  - Reset the TAP
  - Try to get the Device ID by reading the DR
  - If Device ID is 0xFFFFFFFF or if bit 0 != 1, ignore
  - Otherwise, display potentially valid JTAG pinout



# BYPASS Scan

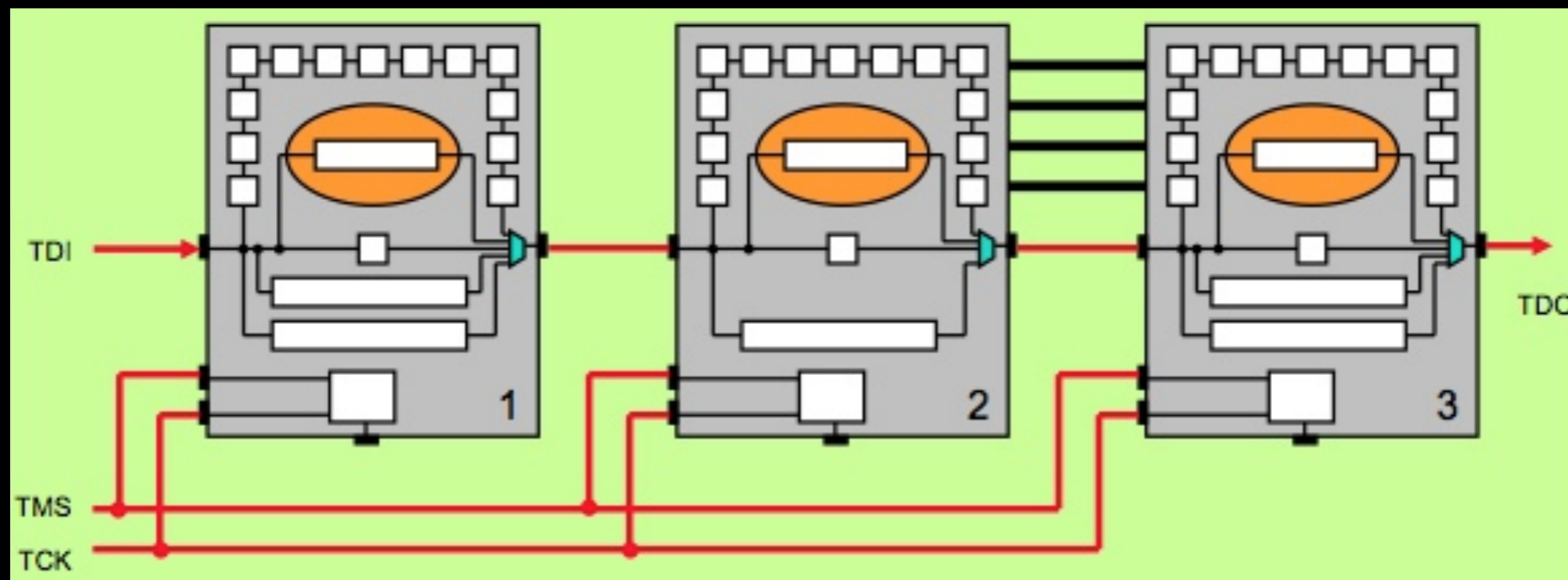
- In BYPASS, data shifted into TDI is received on TDO delayed by one clock cycle





# 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



# BYPASS Scan 3

- Ask user for number of channels to use
- For every possible pin permutation
  - Set unused channels to output high (in case of any active low reset pins)
  - Configure JTAG pins to use on the Propeller
  - Reset the TAP
  - Perform blind interrogation
  - If number of detected devices  $> 0$ , display potentially valid JTAG pinout



# JTAG: Scan Timing

- **IDCODE**
  - TDI ignored since we're only shifting data out of DR
  - ~264 permutations/second
- **BYPASS**
  - Many bits/permutation needed to account for multiple devices in chain and varying IR lengths
  - ~13.37 permutations/second

# of Channels	IDCODE Permutations	IDCODE (mm:ss)	BYPASS Permutations	BYPASS (mm:ss)
4	24	< 00:01	24	00:02
8	336	00:02	1680	02:05
16	3360	00:13	43680	54:27
24	12144	00:46	255024	317:54



# JTAG: Examples



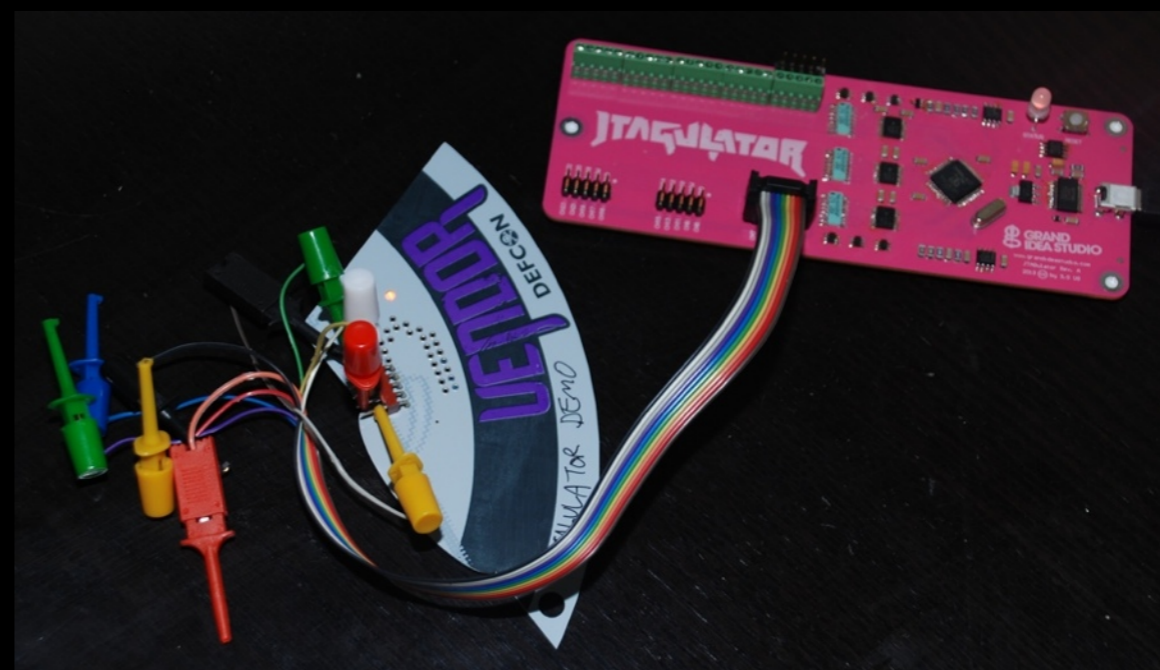
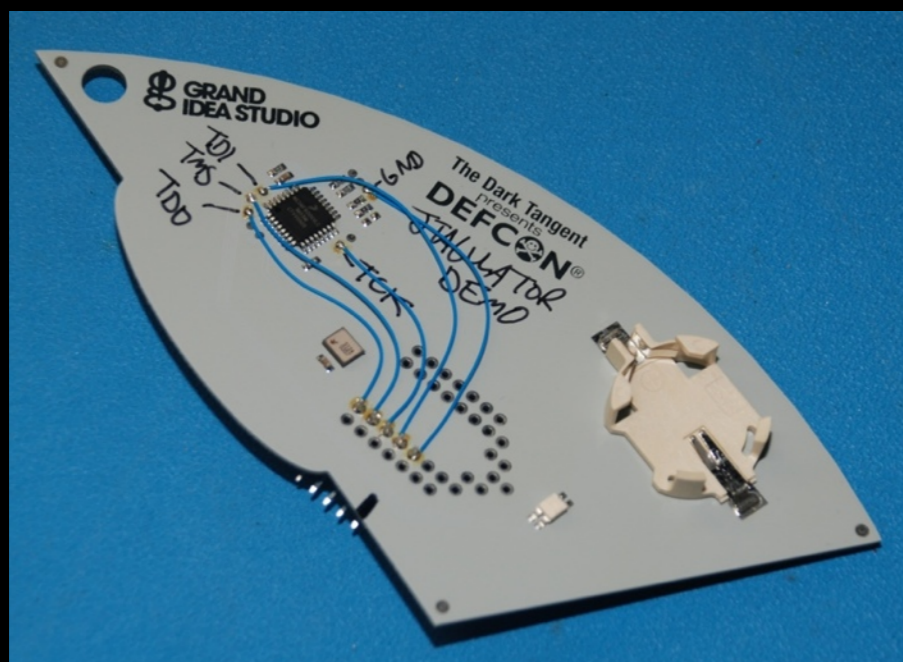
# DEFCON 17 Badge

- Freescale MC56F8006 Digital Signal Controller
  - ID = 0x01C0601D
  - [www.bsdl.info/details.htm?sid=e82c74686c7522e888ca59b002289d77](http://www.bsdl.info/details.htm?sid=e82c74686c7522e888ca59b002289d77)

MSB

LSB

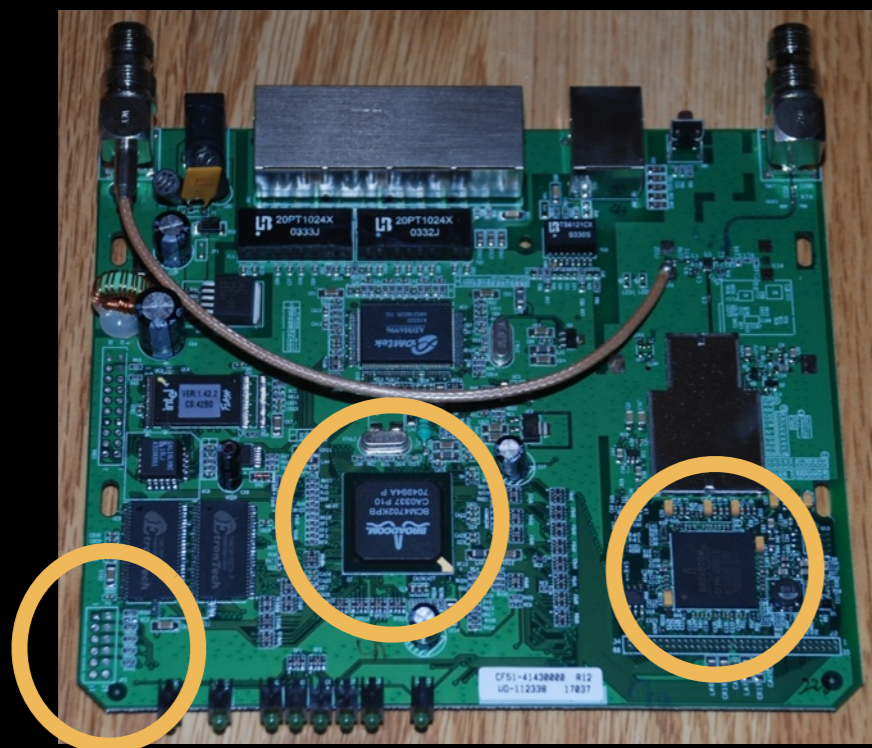
Ver.	Design Center	Core Number	Chip Derivative	Manufacturer ID	Fixed
31...28	27...22	21...17	16...12	11...1	0
0000	000111	00000 (DSP56300)	00110	00000001110 (0x0E)	1



# Linksys WRT54G v1.1

- Broadcom BCM4702 (also contains BCM4306)
  - ID = 0x0471017F
  - <https://github.com/notch/tjtag/blob/master/tjtag.c>

MSB			LSB
Ver.	Part Number	Manufacturer ID	Fixed
31...28	27...12	11...1	0
0000	0100011100010000 (BCM4702 rev. 1)	00010111111 (0xBF)	1



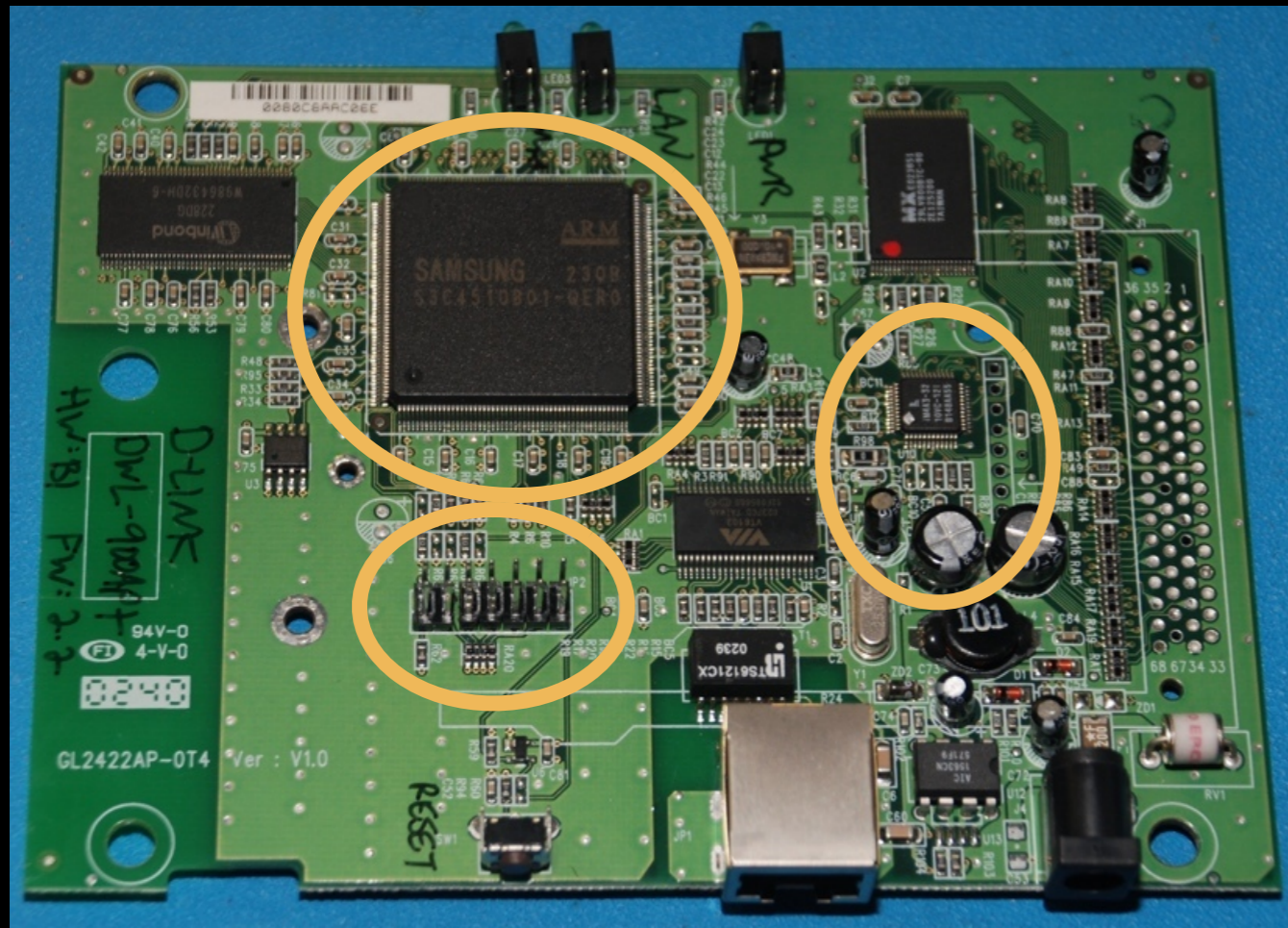
1	nTRST	■ ■	GND	2
3	TDI	■ ■	GND	4
5	TDO	■ ■	GND	6
7	TMS	■ ■	GND	8
9	TCK	■ ■	GND	10
11	nSRST	■ ■	GND	12

\*\*\* [www.jtagtest.com/pinouts/wrt54](http://www.jtagtest.com/pinouts/wrt54)



# D-Link DWL-900AP+

- Samsung S3C4510B01-QER0 CPU (ARM7TDMI)
  - ID = 0x1F0F0F0F
  - <http://pdf1.alldatasheet.com/datasheet-pdf/view/37744/SAMSUNG/S3C4510B.html> (Appendix A)



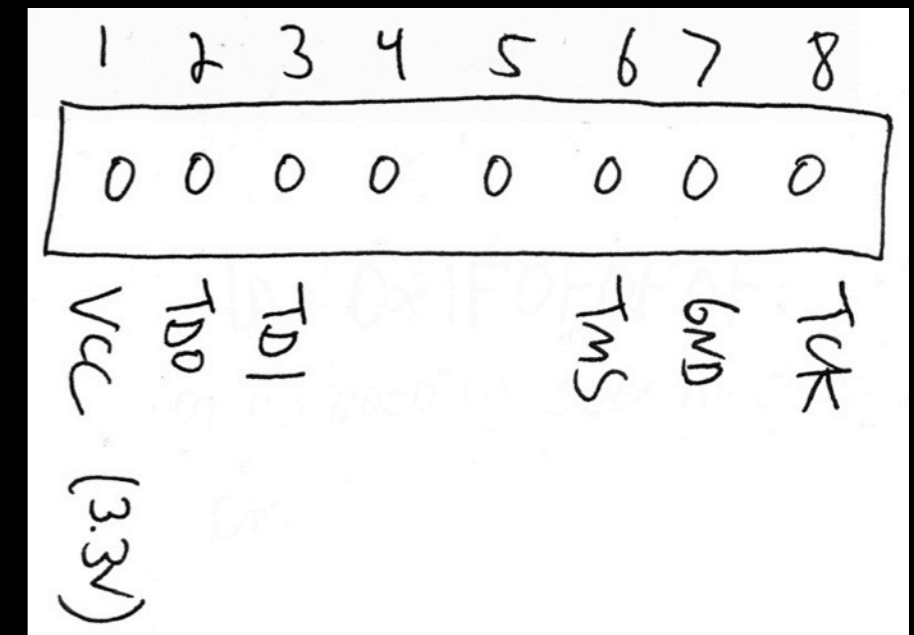
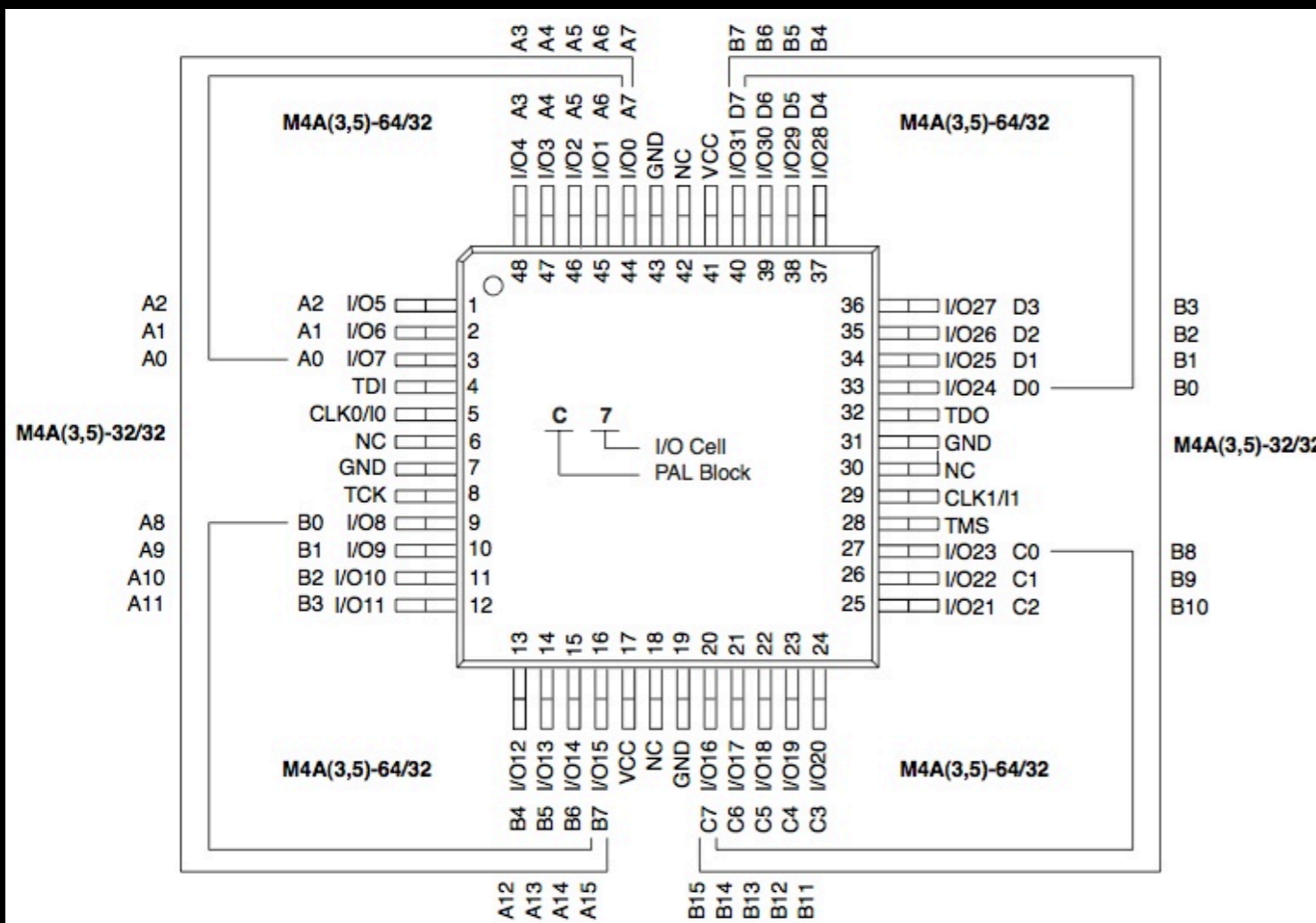
1	VREF	■	■	GND	2
3	nTRST	■	■	GND	4
5	TDI	■	■	GND	6
7	TMS	■	■	GND	8
9	TCK	■	■	GND	10
11	TDO	■	■	nSRST	12
13	VREF	■	■	GND	14

\*\*\* [www.jtagtest.com/pinouts/arm14](http://www.jtagtest.com/pinouts/arm14)



# D-Link DWL-900AP+ 2

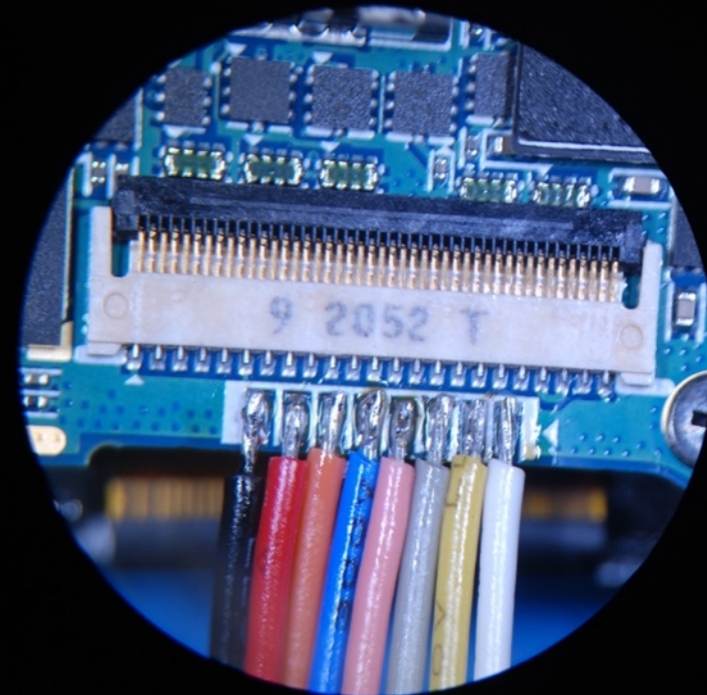
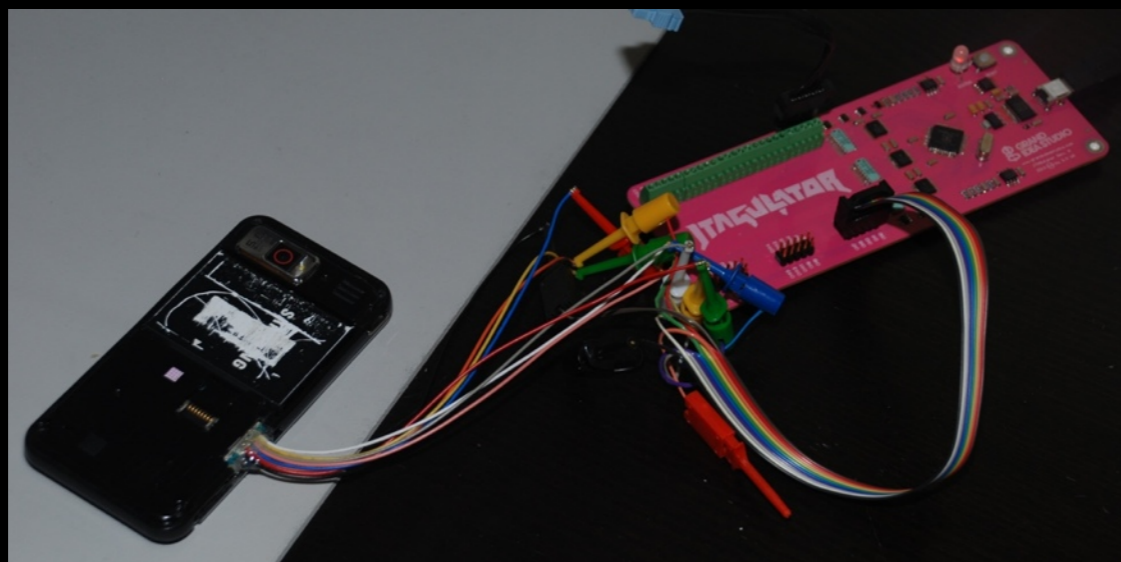
- Lattice ispMACH iM4A3-32 CPLD (TQFP-48)
  - ID = 0x17437157
  - [www.latticesemi.com/lit/docs/bsdl/mach4a3/m4a032t8l\\_isc.bsm](http://www.latticesemi.com/lit/docs/bsdl/mach4a3/m4a032t8l_isc.bsm)





# Samsung SCH-i910

- Marvell PXA312 (Intel XScale/ARM5)
  - ID = 0x2E649013
  - <http://docs.toradex.com/100197-colibri-arm-som-pxa3xx-dm-vol-1.pdf> (Table 9)
  - TDI = 3 (Grey), TMS = 4 (Pink), TCK = 5 (Blue), TDO = 6 (Orange), GND = 8 (Black)
- JTAG disabled when external power supplied or phone is "on" via battery



# 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) =  $1 / \text{bit width}$
  - [http://en.wikipedia.org/wiki/Asynchronous\\_serial\\_communication](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)

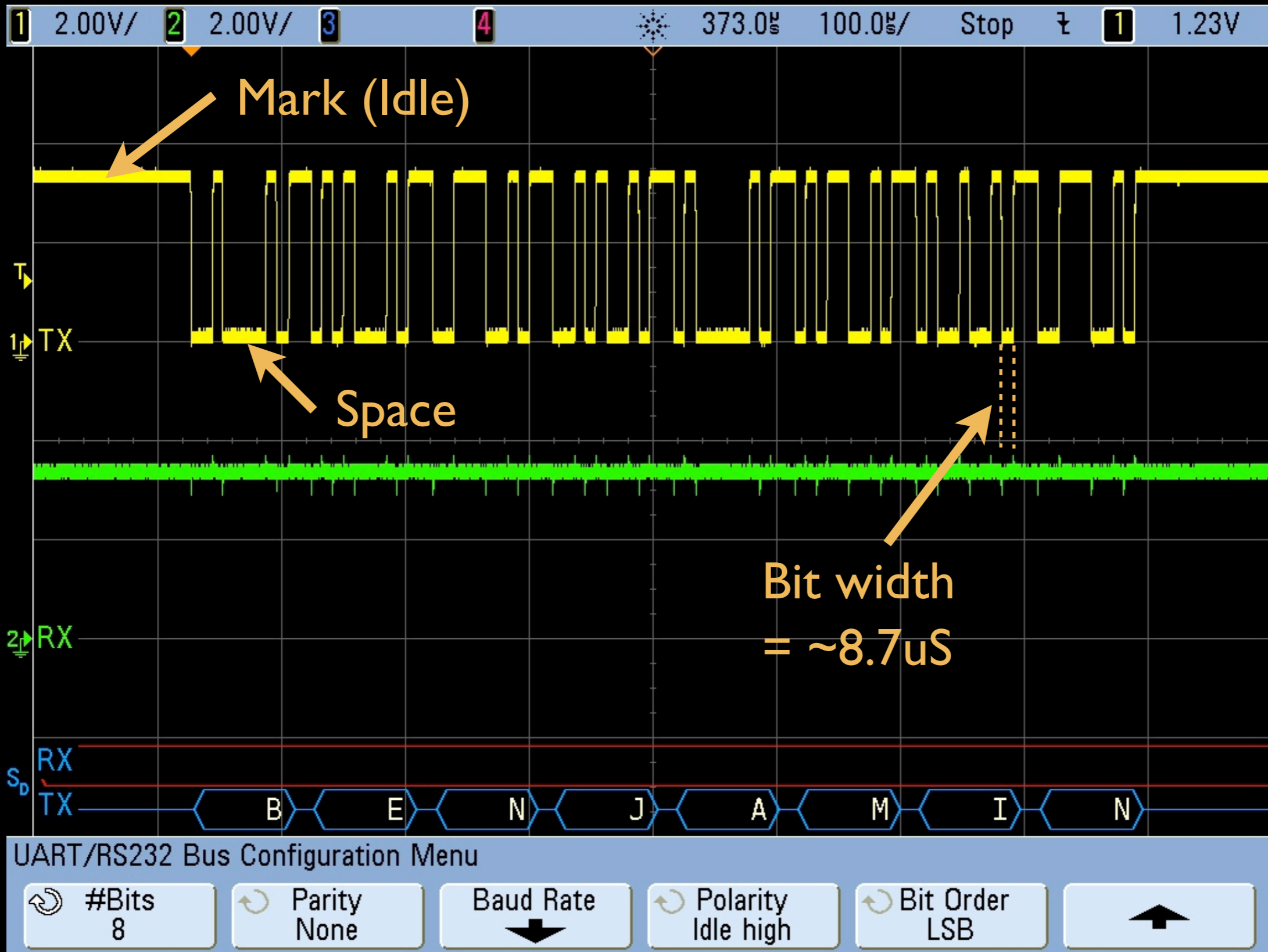


# UART 2

- **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 3



# 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

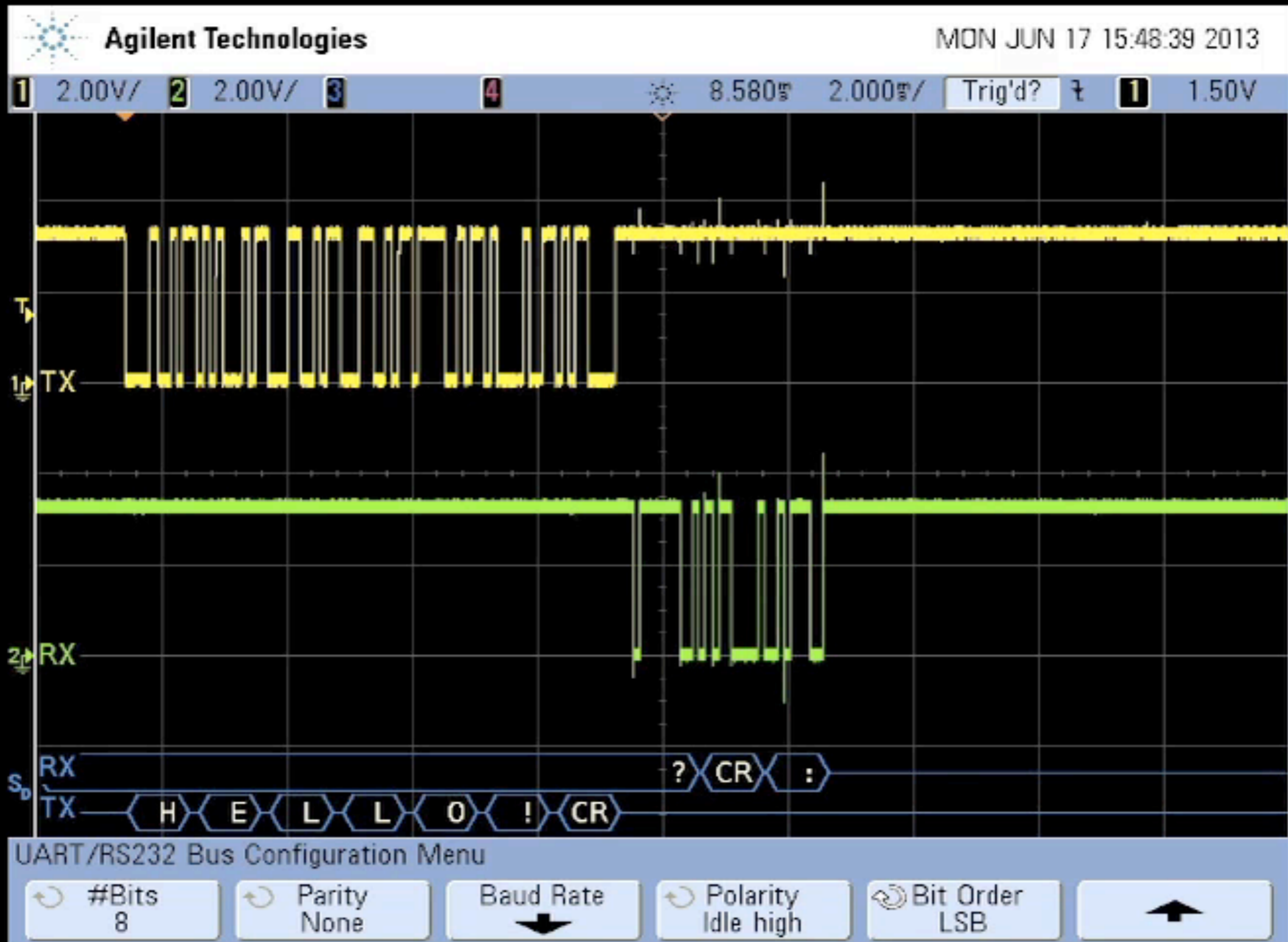


# UART Scan 2

- Ask user for desired output string (up to 16 bytes)
- Ask user for number of channels to use
- For every possible pin permutation
  - Configure UART pins to use on the Propeller
  - Set baud rate
  - Send user string
  - Wait to receive data (20ms maximum per byte)
  - If any bytes received, display potentially valid UART pinout and data (up to 16 bytes)



# UART Scan 3



# UART: Scan Timing

- Only need to locate two pins (TXD/RXD)
- 24 baud rates/permutation
- ~1 permutation/second

# of Channels	UART Permutations	Time (mm:ss)
4	12	00:12
8	56	00:57
16	240	4:04
24	552	9:22



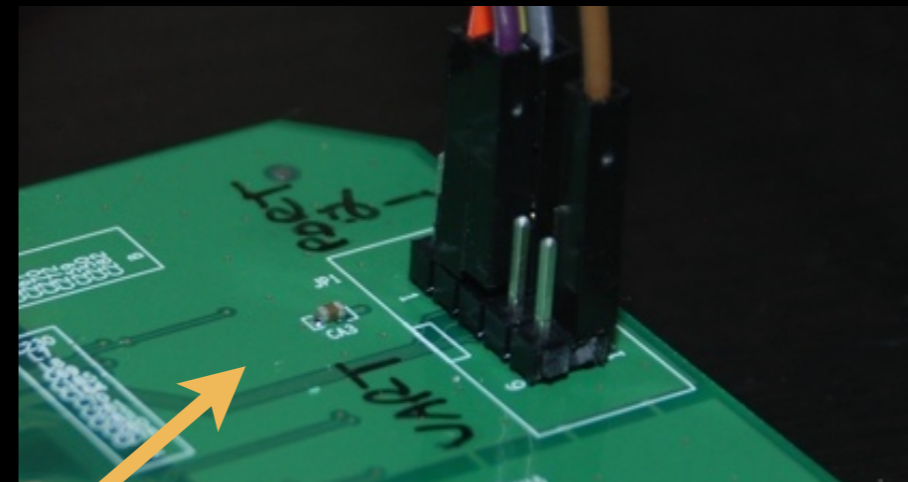
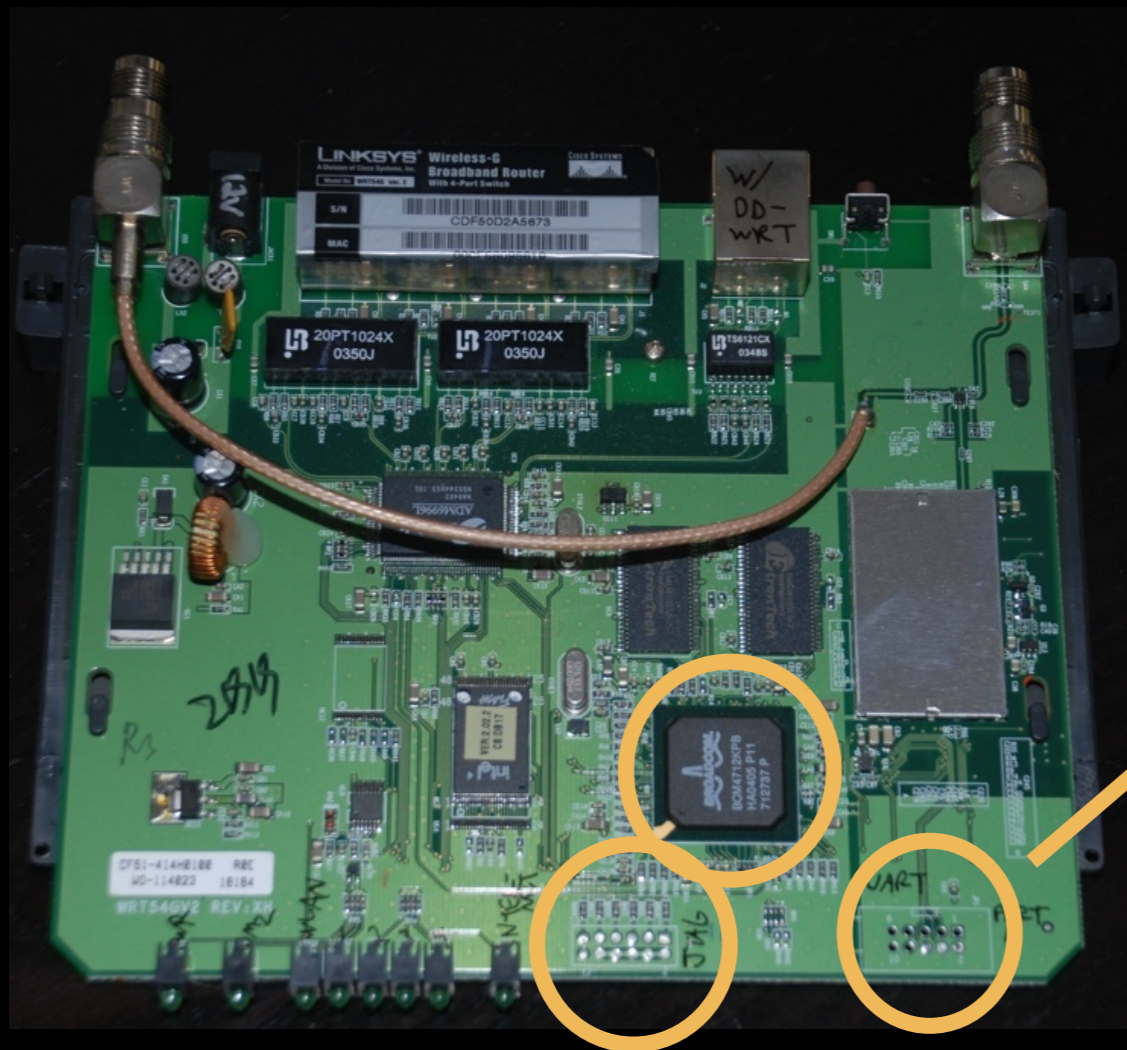


# UART: Examples



# Linksys WRT54G v2 rXH (w/ DD-WRT)

- Broadcom BCM4712
  - ID = 0x1471217F
  - <https://github.com/notch/tjtag/blob/master/tjtag.c>
  - UART: JP1 (TXD = 4, RXD = 6) @ 115200, 8N1



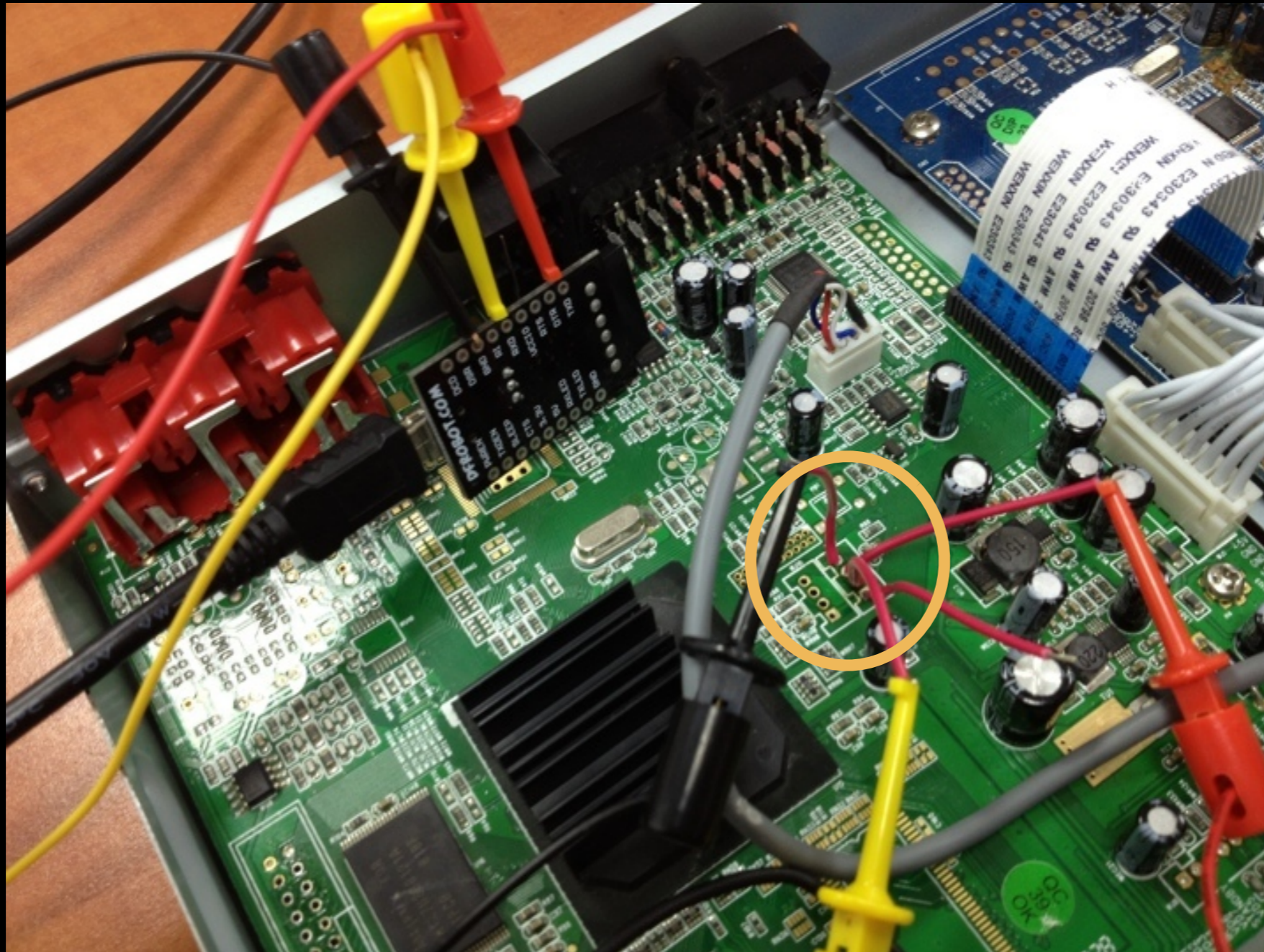
1	nTRST	■	■	GND	2
3	TDI	■	■	GND	4
5	TDO	■	■	GND	6
7	TMS	■	■	GND	8
9	TCK	■	■	GND	10
11	nSRST	■	■	GND	12

\*\*\* [www.jtagtest.com/pinouts/wrt54](http://www.jtagtest.com/pinouts/wrt54)



# Apex STB236 Set Top Box

- Bootloader + U-Boot
  - UART @ 115200, 8N1



# Apex STB236 Set Top Box 2

```
-----  
-- STB222 Lite Primary Bootloader 0.1-3847, NI (04:00:34, Feb 17 2009)  
-- Andre McCurdy, NXP Semiconductors  
-----
```

```
Device: PNX8335 M1  
Secure boot: disabled, keysel: 0, vid: 0 (expecting 2)  
Poly10: 0x00000000  
RNG: enabled  
RSA keyhide: enabled  
UID: 0000000000000000  
AES key: 00000000000000000000000000000000  
KC status: 0x00000000  
Flash config: 7 (omni: 8bit NAND), timing: 0x0C  
CPU clock: 320 MHz  
DRAM: 200 MHz, 1 x 1 64MByte 16bit device (SIF0): 64 MBytes  
NAND: RDY polling disabled  
NAND: (AD76) Hynix SLC, pagesize 512, blocksize 16k, 64 MBytes  
NAND 0x00020000: valid header  
NAND 0x00020000: valid image  
about exec time: 179602 uSec
```

```
U-Boot 1.2.0.dev (Secondary Bootloader) (Jul 31 2009 - 02:53:01)
```

```
CPU: PNX????  
Secure boot: disabled  
DRAM: 64 MB  
NAND: nCS0 (force asserted legacy mode)  
NAND: Hynix 64MiB 3,3V 8-bit  
NAND 0x02a3c000: bad block  
NAND 0x030bc000: bad block  
NAND 0x03478000: bad block  
NAND 0x0385c000: bad block  
Board Opts: SCART PAL  
Splash: done  
u-boot startup time so far: 1012 msec  
Hit any key to stop autoboot: 1 ... 0
```

```
STB225v1 nand#
```



# Demonstrations



# 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 (/TRST)
  - 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](http://www.jtagulator.com)  
\*\*\* Schematics, source code, BOM, block diagram, Gerber plots, photos, other engineering documentation
- [www.parallax.com](http://www.parallax.com)  
\*\*\* Assembled units, accessories
- <http://oshpark.com/profiles/joegrand>  
\*\*\* Bare boards





# Hands On

- Experiment w/ target devices
- Install Propeller tools
- Firmware walkthrough/review
- Modify firmware
- ???



The End.

