Assisted Discovery of On-Chip Debug Interfaces
Joe Grand (@joegrand)
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

- **JTAGenum & RS232enum**
  - http://deadhacker.com/tools/

- **DARPA Cyber Fast Track**
  - 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/

← 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
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
Host PC
USB Mini-B

Serial-to-USB
FT232RL

1.2V - 3.3V
~13mV/step

D/A
AD8655

Power Switch
MIC2025-2YM

MIC2025-2YM
Parallax Propeller

Voltage Level Translator
TXS0108EPWR

Voltage Level Translator
TXS0108EPWR

Voltage Level Translator
TXS0108EPWR

Input Protection Circuitry

Target Device

USB
5V

5V

3.3V

3.3V

1 (PWM)

2

2 (I2C)

WP59EGW

Status Indicator

MCU

EEPROM
24LC512

Voltage Level Translator
TXS0108EPWR

LOD
LD1117S33TR

USB
5V

3.3V

17S33TR

AD8655

Host PC
USB Mini-B

Serial-to-USB
FT232RL

1.2V - 3.3V
~13mV/step

Serial-to-USB
FT232RL

1.2V - 3.3V
~13mV/step
PCB

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

Target I/F (24 channels)

Input protection

Level translation

Status

Propeller

USB

Op-Amp/DAC
Propeller/Core

- Completely custom, ground up design
- 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/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
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 @ Vo = 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 (40kΩ when driving low, 4kΩ when high)
  – Automatic signal direction detection
  – High–Z outputs when OE low -> will not interfere with target when not in use
Level Translation 2

VCCA <= VCCB
VCCA range: 1.2V to 3.6V
VCCB range: 1.7V to 5.5V
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**
  - $V_f$ @ 1mA = 0.2V typ., 0.35V max.
  - $V_f$ @ 10mA = 0.25V typ., 0.45V max.
  - Alternate: SD103ASDM
Bill-of-Materials

<table>
<thead>
<tr>
<th>Item</th>
<th>Quantity</th>
<th>Reference</th>
<th>Manufacturer</th>
<th>Manuf. Part #</th>
<th>Distributor</th>
<th>Distrib. Part #</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>C1, C2</td>
<td>Kemet</td>
<td>C1206C103K5RACTU</td>
<td>Digi-Key</td>
<td>399-1234-1-ND</td>
<td>Capacitor, 0.01uF ceramic, 10%, 50V, X7R, 1206</td>
</tr>
<tr>
<td>2</td>
<td>14</td>
<td>C3, C6, C9, C11, C12, C13, C14, C15, C17, C18, C19, C20, C21, C22</td>
<td>Kemet</td>
<td>C1206C104K5RACTU</td>
<td>Digi-Key</td>
<td>399-1249-1-ND</td>
<td>Capacitor, 0.1uF ceramic, 10%, 50V, X7R, 1206</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>C4</td>
<td>Yageo</td>
<td>CCA1206KRX7FRBB102</td>
<td>Digi-Key</td>
<td>311-1170-1-ND</td>
<td>Capacitor, 1000pF ceramic, 10%, 50V, X7R, 1206</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>C5</td>
<td>Yageo</td>
<td>CCA1206KRX7FRBB471</td>
<td>Digi-Key</td>
<td>311-1167-1-ND</td>
<td>Capacitor, 470pF ceramic, 10%, 50V, X7R, 1206</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>C7</td>
<td>Kemet</td>
<td>T491A106M016AS</td>
<td>Digi-Key</td>
<td>399-3687-1-ND</td>
<td>Capacitor, 10uF tantalum, 20%, 16V, size A</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>C8, C10</td>
<td>Kemet</td>
<td>T491A475K016AT</td>
<td>Digi-Key</td>
<td>399-3697-1-ND</td>
<td>Capacitor, 4.7uF tantalum, 10%, 16V, size A</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>D1</td>
<td>Kingbright</td>
<td>WP59EGW</td>
<td>Digi-Key</td>
<td>754-1232-1-ND</td>
<td>LED, Red/Green Bi-Color, T-1 3/4 (5mm)</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>L1</td>
<td>TDK</td>
<td>MPZ2012S221A</td>
<td>Digi-Key</td>
<td>445-1568-1-ND</td>
<td>Inductor, Ferrite Bead, 220@100MHz, 3A, 0805</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>P1</td>
<td>Hirose Electric</td>
<td>UX60-MB-5S8</td>
<td>Digi-Key</td>
<td>H2960CT-ND</td>
<td>Connector, Mini-USB, 5-pin, SMT w/ PCB mount</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>P2, P3, P4, P5, P6</td>
<td>TE Connectivity</td>
<td>262834-5</td>
<td>Digi-Key</td>
<td>A93836-ND</td>
<td>Connector, Terminal Block, 5-pin, side entry, 0.1&quot; P</td>
</tr>
<tr>
<td>11</td>
<td>3</td>
<td>P7, P8, P9</td>
<td>SM</td>
<td>666210-6404-AR</td>
<td>Digi-Key</td>
<td>3M9460-ND</td>
<td>Header, Dual row, Vertical header, 2x5-pin, 0.1&quot; P</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>Q1</td>
<td>Fairchild</td>
<td>MMBT3904</td>
<td>Digi-Key</td>
<td>MMBT3904FSCT-ND</td>
<td>Transistor, NPN, 40V, 200mA, SOT23-3</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>R1, R2, R3, R4, R10</td>
<td>Any</td>
<td>Any</td>
<td>Digi-Key</td>
<td>P10KECT-ND</td>
<td>Resistor, 10k, 5%, 1/4W, 1206</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>R5</td>
<td>Any</td>
<td>Any</td>
<td>Digi-Key</td>
<td>P470ECT-ND</td>
<td>Resistor, 470 ohm, 5%, 1/4W, 1206</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>R6</td>
<td>Any</td>
<td>Any</td>
<td>Digi-Key</td>
<td>P270ECT-ND</td>
<td>Resistor, 270 ohm, 5%, 1/4W, 1206</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td>R7</td>
<td>Any</td>
<td>Any</td>
<td>Digi-Key</td>
<td>P18.0KFCT-ND</td>
<td>Resistor, 18k, 1%, 1/4W, 1206</td>
</tr>
<tr>
<td>17</td>
<td>1</td>
<td>R8</td>
<td>Any</td>
<td>Any</td>
<td>Digi-Key</td>
<td>P8.20KFCT-ND</td>
<td>Resistor, 8.2k, 1%, 1/4W, 1206</td>
</tr>
<tr>
<td>18</td>
<td>1</td>
<td>R9</td>
<td>Any</td>
<td>Any</td>
<td>Digi-Key</td>
<td>P100KECT-ND</td>
<td>Resistor, 100k, 5%, 1/4W, 1206</td>
</tr>
<tr>
<td>19</td>
<td>3</td>
<td>R11, R12, R13</td>
<td>Bourns</td>
<td>4816P-1-102LF</td>
<td>Digi-Key</td>
<td>4816P-1-102LFCT-ND</td>
<td>Resistor, Array, 8 isolated, 1k 2%, 1/6W, SOIC16</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>SW1</td>
<td>C&amp;S</td>
<td>KSC201JFS</td>
<td>Digi-Key</td>
<td>401-1756-1-ND</td>
<td>Switch, SPST, Momentary, 120gf, 6.2 x 6.2mm, J-Lead</td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>U1</td>
<td>FTDI</td>
<td>FT232RL-REEL</td>
<td>Digi-Key</td>
<td>768-1007-1-ND</td>
<td>IC, USB-to-UART Bridge, SSOP28</td>
</tr>
<tr>
<td>22</td>
<td>1</td>
<td>U2</td>
<td>Parallax</td>
<td>P8X32A-Q44</td>
<td>Digi-Key</td>
<td>P8X32A-Q44ND</td>
<td>IC, Microcontroller, Propeller, LQFP44</td>
</tr>
<tr>
<td>23</td>
<td>1</td>
<td>U3</td>
<td>Micrel</td>
<td>MIC2025-2YM</td>
<td>Digi-Key</td>
<td>576-1058-1-ND</td>
<td>IC, Power Distribution Switch, Single-channel, SOIC8</td>
</tr>
<tr>
<td>24</td>
<td>1</td>
<td>U4</td>
<td>Microchip</td>
<td>24LC512-I/SN</td>
<td>Digi-Key</td>
<td>24LC512-I/SN-ND</td>
<td>IC, Memory, Serial EEPROM, 64KB, SOIC8</td>
</tr>
<tr>
<td>25</td>
<td>1</td>
<td>U5</td>
<td>Analog Devices</td>
<td>AD8655ARZ</td>
<td>Digi-Key</td>
<td>AD8655ARZ-ND</td>
<td>IC, Op Amp, CMOS, Rail-to-rail, 220mA Iout, SOIC8</td>
</tr>
<tr>
<td>26</td>
<td>1</td>
<td>U6</td>
<td>ST Microelectronics</td>
<td>LDT117S33CTR</td>
<td>Digi-Key</td>
<td>497-1241-1-ND</td>
<td>IC, Voltage Regulator, LDO, 5.3V@800mA, SOT223</td>
</tr>
<tr>
<td>27</td>
<td>6</td>
<td>U7, U8, U10, U11, U13, U14</td>
<td>ON Semiconductor</td>
<td>NUP4302MR6T1G</td>
<td>Digi-Key</td>
<td>NUP4302MR6T1GOSCT-ND</td>
<td>IC, Schottky Diode Array, 4 channel, TSOP6</td>
</tr>
<tr>
<td>28</td>
<td>3</td>
<td>U9, U12, U15</td>
<td>Texas Instruments</td>
<td>TXS0108EPWR</td>
<td>Digi-Key</td>
<td>296-23011-1-ND</td>
<td>IC, Level Translator, Bi-directional, TSSOP20</td>
</tr>
<tr>
<td>29</td>
<td>1</td>
<td>Y1</td>
<td>ECS</td>
<td>ECS-50-18-4XEN</td>
<td>Digi-Key</td>
<td>XC1738-ND</td>
<td>Crystal, 5.0MHz, 18pF, HC49/U3</td>
</tr>
<tr>
<td>30</td>
<td>1</td>
<td>PCB</td>
<td>Any</td>
<td>JTAG B</td>
<td>N/A</td>
<td>N/A</td>
<td>PCB, Fabrication</td>
</tr>
</tbody>
</table>
Firmware
Source Tree

JTAGulator.spin
  ├── Parallax Serial Terminal.spin
  │     └── RealRandom.spin
  │        └── PropJTAG.spin
  │             └── JDCogSerial.spin
Propeller Resources

RAM Usage

<table>
<thead>
<tr>
<th>$0010</th>
<th>Program</th>
<th>1,870 Longs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Variable</td>
<td>76 Longs</td>
</tr>
<tr>
<td></td>
<td>Stack / Free</td>
<td>6,242 Longs</td>
</tr>
</tbody>
</table>

Clock Mode: XTAL1 + PLL16X
Clock Freq: 80,000,000 Hz
XIN Freq: 5,000,000 Hz
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)

• 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
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: 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 with TMS high for 5x TCK
- 3 primary steps in Scan: Capture, Shift, Update
- Data held in "shadow" latch until Update state
## JTAG: Instructions

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

• Flyswatter2
  – www.tincantools.com

• Bus Blaster (open source)
  – http://dangerousprototypes.com/docs/Bus_Blaster

• Wiggler or compatible (parallel port)
JTAG: SW Tools

- OpenOCD (Open On-Chip Debugger)

- 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
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
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

<table>
<thead>
<tr>
<th># of Channels</th>
<th>IDCODE Permutations</th>
<th>IDCODE (mm:ss)</th>
<th>BYPASS Permutations</th>
<th>BYPASS (mm:ss)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>24</td>
<td>&lt; 00:01</td>
<td>24</td>
<td>00:02</td>
</tr>
<tr>
<td>8</td>
<td>336</td>
<td>00:02</td>
<td>1680</td>
<td>02:05</td>
</tr>
<tr>
<td>16</td>
<td>3360</td>
<td>00:13</td>
<td>43680</td>
<td>54:27</td>
</tr>
<tr>
<td>24</td>
<td>12144</td>
<td>00:46</td>
<td>255024</td>
<td>317:54</td>
</tr>
</tbody>
</table>
JTAG: Examples
DEFCON 17 Badge

- Freescale MC56F8006 Digital Signal Controller
  - ID = 0x01C0601D
  - www.bsdsl.info/details.htm?sid=e82c74686c7522e888ca59b002289d77

<table>
<thead>
<tr>
<th>MSB</th>
<th>LSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ver.</td>
<td>Design Center</td>
</tr>
<tr>
<td>31...28</td>
<td>27...22</td>
</tr>
<tr>
<td>0000</td>
<td>000111</td>
</tr>
</tbody>
</table>
Linksys WRT54G v1.1

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

<table>
<thead>
<tr>
<th>MSB</th>
<th>Part Number</th>
<th>Manufacturer ID</th>
<th>Fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>31...28</td>
<td>27...12</td>
<td>11...1</td>
<td>0</td>
</tr>
<tr>
<td>0000</td>
<td>0100011100010000 (BCM4702 rev. 1)</td>
<td>0001011111 (0xBF)</td>
<td>1</td>
</tr>
</tbody>
</table>

*** www.jtagtest.com/pinouts/wrt54
D-Link DWL-900AP+

- Samsung S3C4510B01-QER0 CPU (ARM7TDMI)
  - ID = 0x1F0F0F0F

*** 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
Samsung SCH-i910

- **Marvell PXA312 (Intel XScale/ARM5)**
  - ID = 0x2E649013
  - 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 / bit width

*** 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

Mark (Idle)

Space

Bit width = ~8.7 μS
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

<table>
<thead>
<tr>
<th># of Channels</th>
<th>UART Permutations</th>
<th>Time (mm:ss)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>12</td>
<td>00:12</td>
</tr>
<tr>
<td>8</td>
<td>56</td>
<td>00:57</td>
</tr>
<tr>
<td>16</td>
<td>240</td>
<td>4:04</td>
</tr>
<tr>
<td>24</td>
<td>552</td>
<td>9:22</td>
</tr>
</tbody>
</table>
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

*** 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: 0x0000000000000000
RNG: enabled
RSA keyhide: enabled
UID: 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
aboot 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
  *** Schematics, source code, BOM, block diagram, Gerber plots, photos, other engineering documentation

• 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.