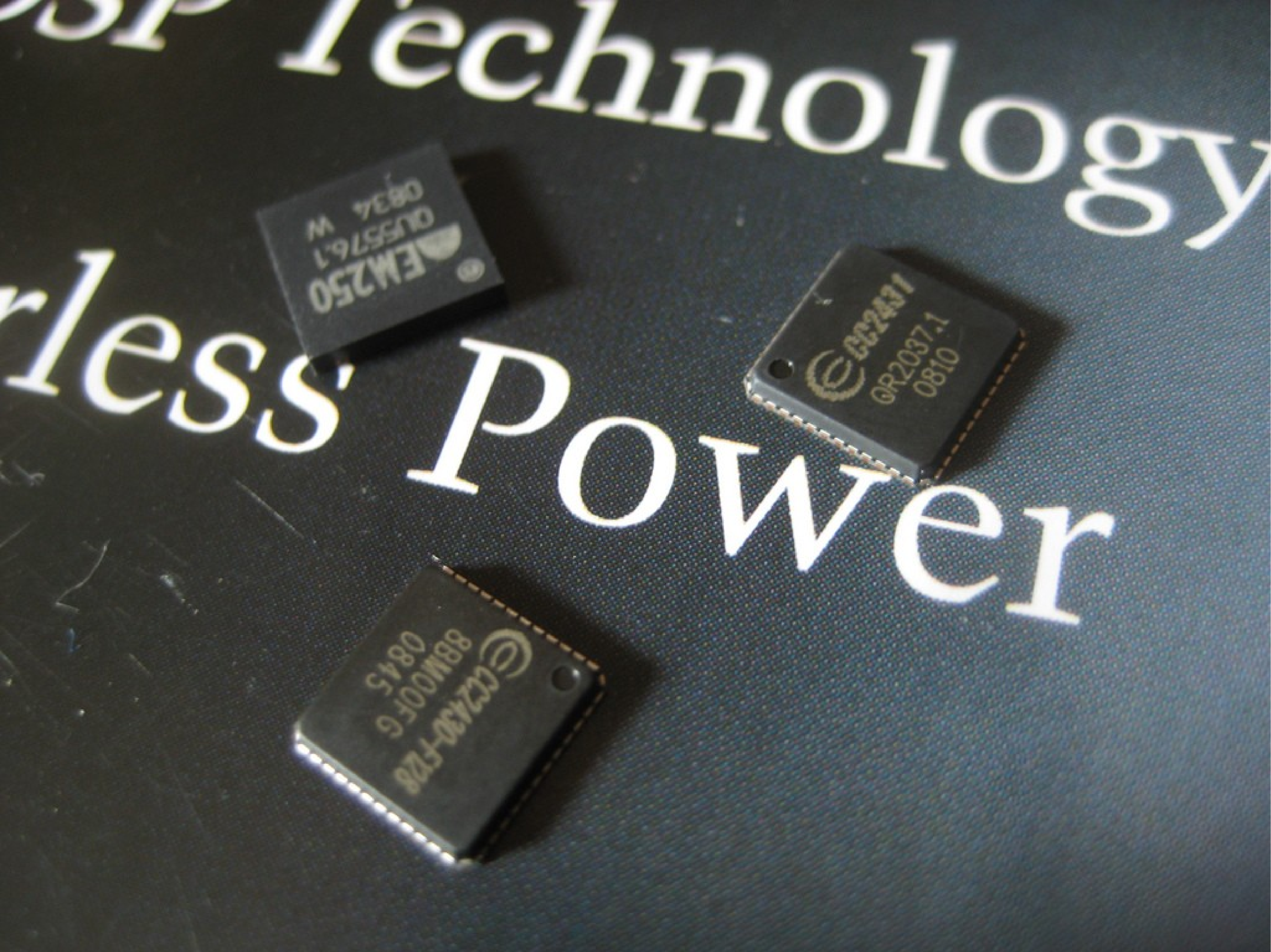
A 16 Bit Rootkit, and Second Generation Zigbee Chips

Travis Goodspeed
travis@radiantmachines.com

Black Hat USA, 2009
Las Vegas, NV


Technology
rless Power
EM250
0U5576.1 W
0834
CC2431
QR2037.1
0810
CC2430-F128
88M00F6
0845

Topics for Today

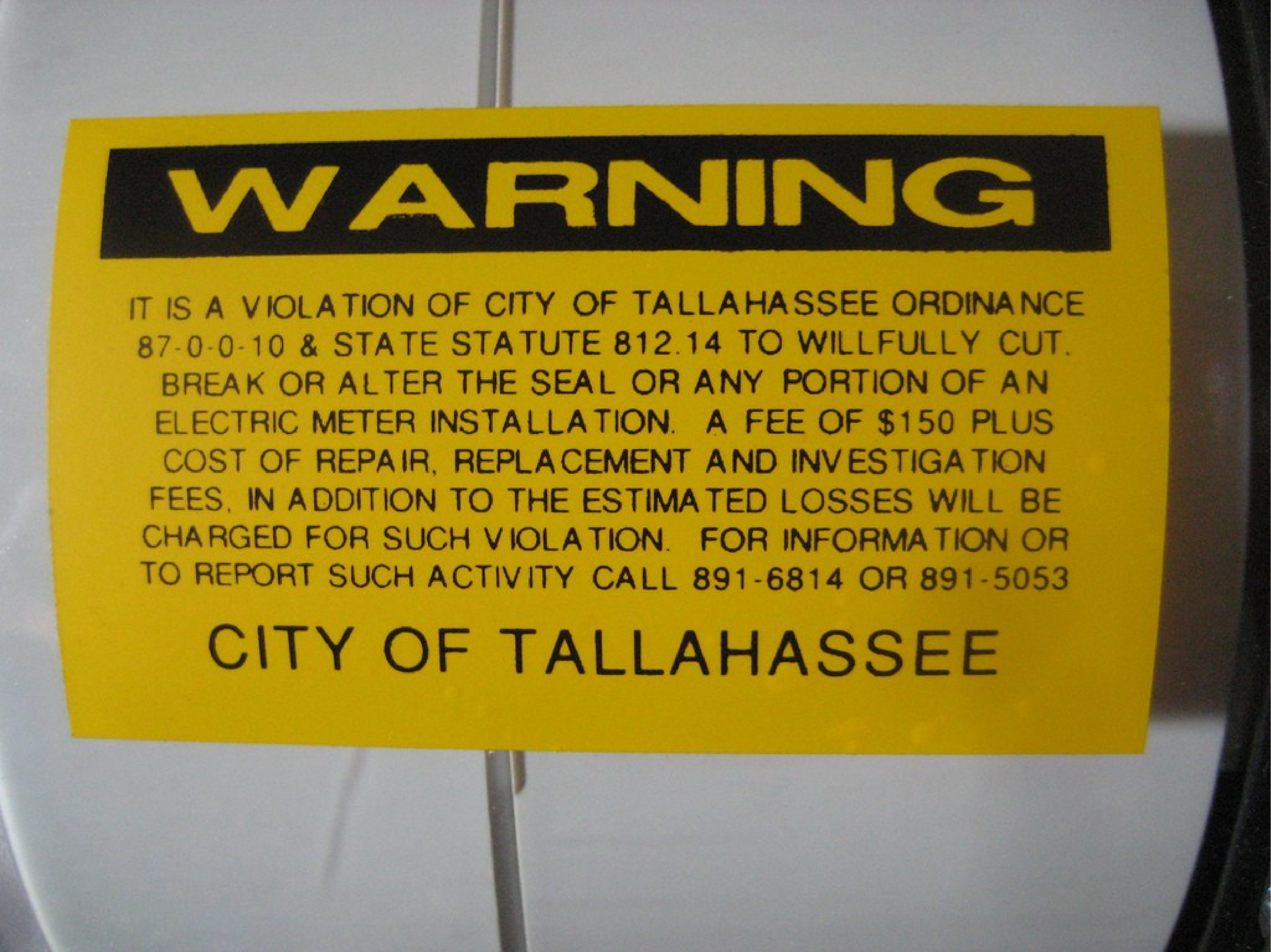
- Second Generation Zigbee Chips
  - EM250, CC2430, CC2530
  - How to break them.
- A 16 Bit Rootkit
  - A very portable operating system,
  - easily injected into a µC application,
  - without damaging that application.

Notice That

- In IT,
  - Malware is common.
  - It's annoying.
  - Simple malware is detected, removed.
- In embedded systems,
  - Malware is rare.
  - No one looks for it.
  - Simple malware is undetected, sufficient.

Forward

- Confidentiality
  - Only to prevent plagiarism.
- Integrity
  - Only against accidental corruption.
- Availability
  - A watchdog timer.



WARNING

IT IS A VIOLATION OF CITY OF TALLAHASSEE ORDINANCE 87-0-0-10 & STATE STATUTE 812.14 TO WILLFULLY CUT. BREAK OR ALTER THE SEAL OR ANY PORTION OF AN ELECTRIC METER INSTALLATION. A FEE OF $150 PLUS COST OF REPAIR, REPLACEMENT AND INVESTIGATION FEES, IN ADDITION TO THE ESTIMATED LOSSES WILL BE CHARGED FOR SUCH VIOLATION. FOR INFORMATION OR TO REPORT SUCH ACTIVITY CALL 891-6814 OR 891-5053

CITY OF TALLAHASSEE

In this Episode

- EM250
  - WTF were they thinking?
- CC2430/CC2530
  - Keys are easily extracted.
- MSP430
  - A rootkit design.
  - How to recognize one, or to build one.

Disclaimers

- EM250/260
  - EM3xx will be better.
- CC2430/CC2530
  - CC430 will be better.
- MSP430
  - MSP430 only chosen for a concrete example.

Brief Review: Microcontrollers

- Little computer.
  - 8 or 16 bit
  - Von Neumann or Harvard
  - Internal Flash/RAM
  - No/partial MMU
- Still a computer.

Brief Review: Wireless Sensors

- Radio+MCU=WSN
- Ultra low power, long deployment.
- Mesh Networking
- Applications
  - Smart Grid
  - Military
  - Wildlife, Geological Research

Brief Review: Terms

- 802.15.4, MAC and lower layers.
- Zigbee, upper layers.
- MSP430, a 16 bit µC
- First Gen Radios, just a radio
- Second Gen Radios, radio+µC

Part 1:
Second Generation Zigbee Chips

Travis Goodspeed
Black Hat 09

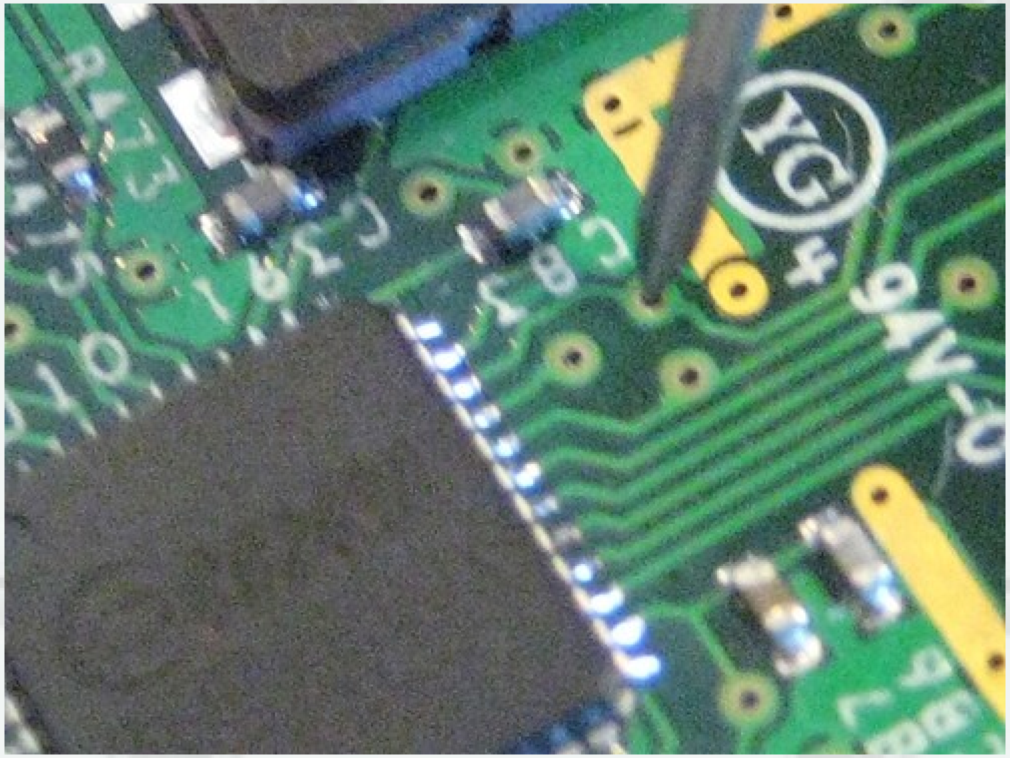
First Generation

- CC2420, EM2420
  - Same chip!
- Just a radio.
  - Keys are sent by SPI.
  - As cleartext.

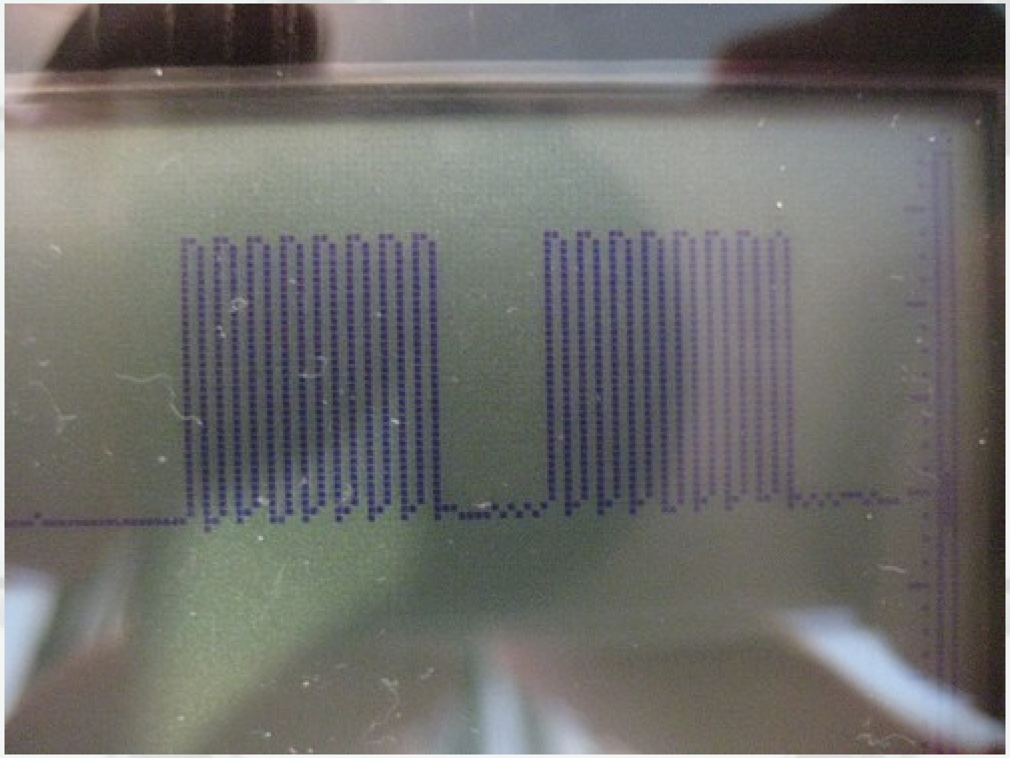
Zigbee Bus Snooping

- First presented at S4 Miami.
  - Later Source Boston, HackADay.
  - Workshop at Defcon!
- Dirt simple,
  - Stick needles into the board's test points.
  - Capture SPI traffic live.
  - Read the AES128 key.
  - Set your radio to the same.

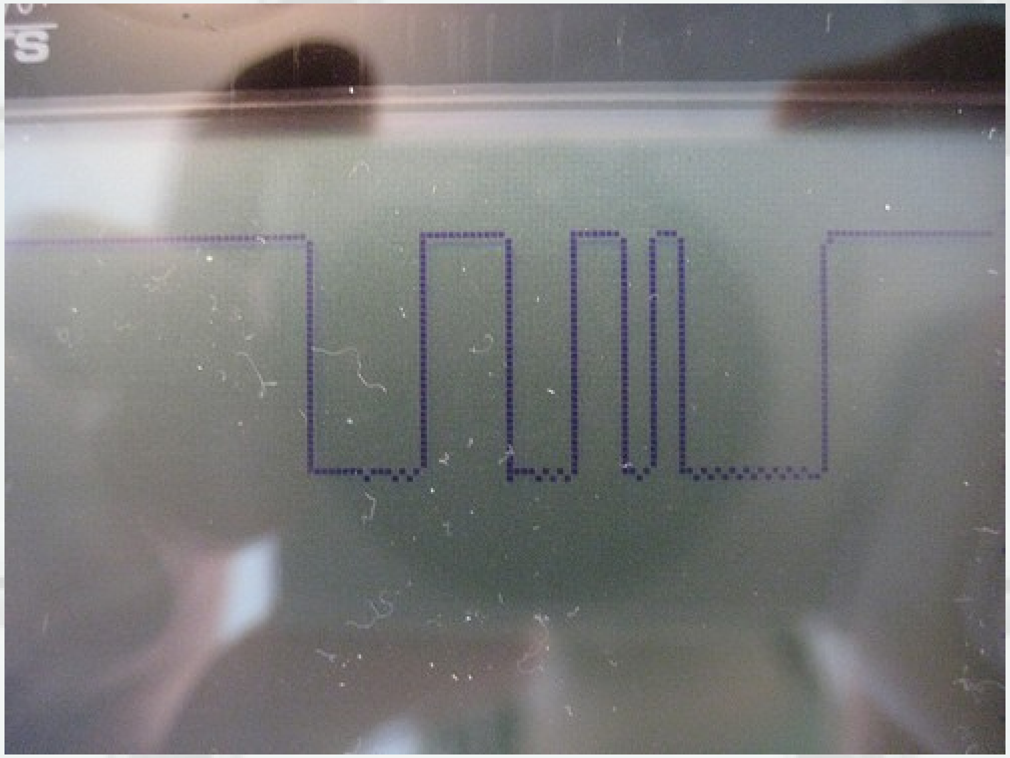
Bus Snooping: Needles



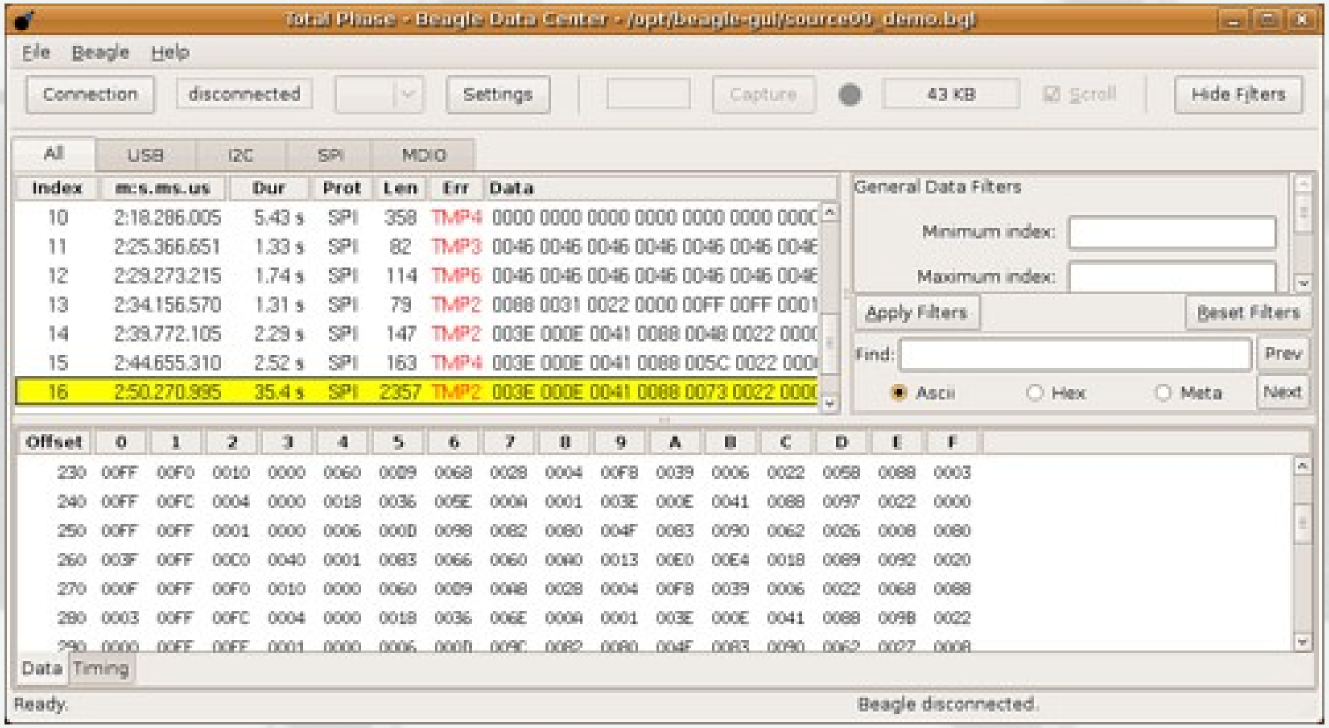
Bus Snooping: Scope



Bus Snooping: Scope



Bus Snooping: Sniffer



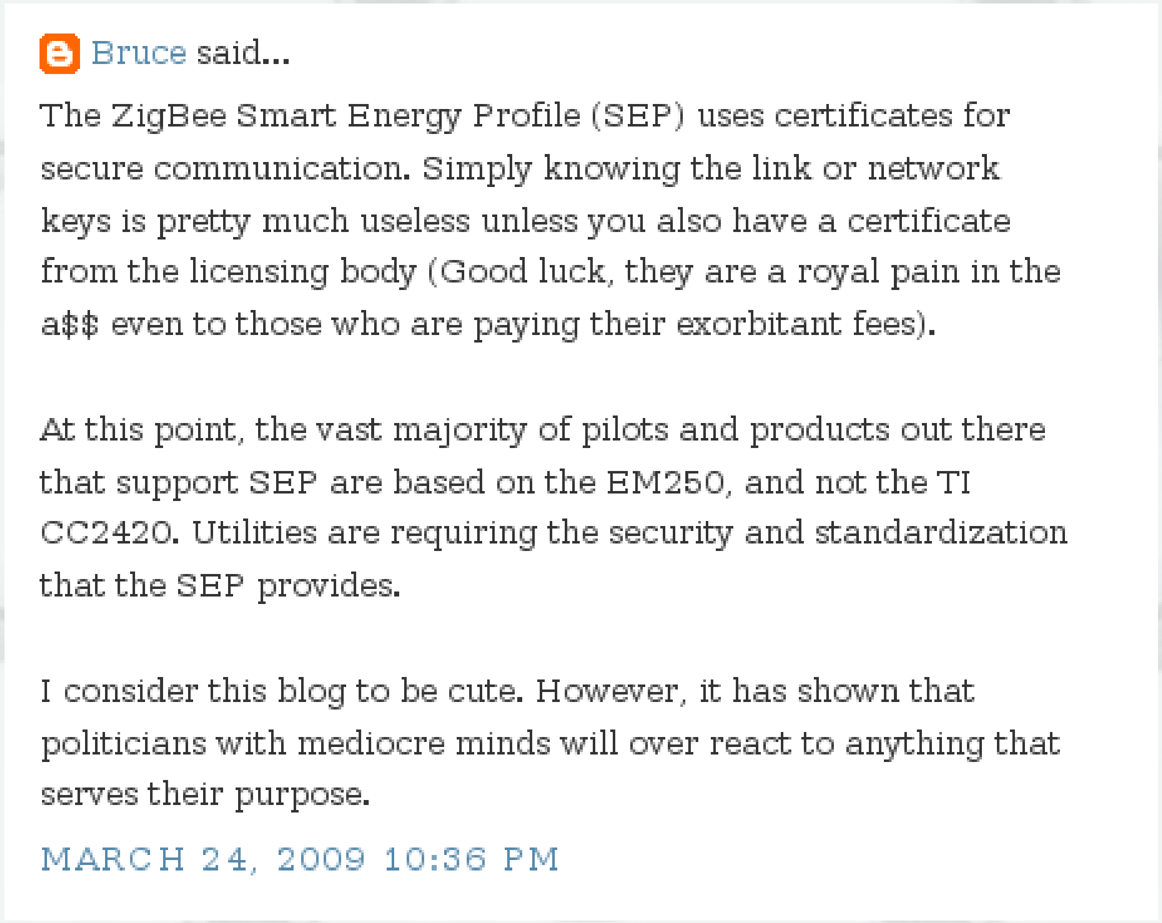
Index	m:s.ms.us	Dur	Prot	Len	Err	Data
10	2:18.286.005	5.43 s	SPI	358	TMP4	0000 0000 0000 0000 0000 0000 000C
11	2:25.366.651	1.33 s	SPI	82	TMP3	0046 0046 0046 0046 0046 0046 004E
12	2:29.273.215	1.74 s	SPI	114	TMP6	0046 0046 0046 0046 0046 0046 004E
13	2:34.156.570	1.31 s	SPI	79	TMP2	0088 0031 0022 0000 00FF 00FF 0001
14	2:39.772.105	2.29 s	SPI	147	TMP2	003E 000E 0041 0088 0048 0022 000C
15	2:44.655.310	2.52 s	SPI	163	TMP4	003E 000E 0041 0088 005C 0022 000
16	2:50.270.995	35.4 s	SPI	2357	TMP2	003E 000E 0041 0088 0073 0022 000C

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
230	00FF	00F0	0010	0000	0060	0009	0068	0028	0004	00F8	0039	0006	0022	0058	0088	0003
240	00FF	00FC	0004	0000	0018	0036	005E	000A	0001	003E	000E	0041	0088	0097	0022	0000
250	00FF	00FF	0001	0000	0006	000D	0098	0082	0080	004F	0083	0090	0062	0026	0008	0080
260	003F	00FF	00C0	0040	0001	0083	0066	0060	00A0	0013	00E0	00E4	0018	0089	0092	0020
270	000F	00FF	00F0	0010	0000	0060	0009	00A8	0028	0004	00F8	0039	0006	0022	0068	0088
280	0003	00FF	00FC	0004	0000	0018	0036	006E	000A	0001	003E	000E	0041	0088	009B	0022
290	0000	00FF	00FF	0001	0000	0006	000D	009C	0082	0080	004F	0083	0090	0062	0027	0008

Ready.

Beagle disconnected.

People are mean!



Bruce said...

The ZigBee Smart Energy Profile (SEP) uses certificates for secure communication. Simply knowing the link or network keys is pretty much useless unless you also have a certificate from the licensing body (Good luck, they are a royal pain in the a$$ even to those who are paying their exorbitant fees).

At this point, the vast majority of pilots and products out there that support SEP are based on the EM250, and not the TI CC2420. Utilities are requiring the security and standardization that the SEP provides.

I consider this blog to be cute. However, it has shown that politicians with mediocre minds will over react to anything that serves their purpose.

MARCH 24, 2009 10:36 PM

Again

- “...the vast majority of pilots and products out there that support SEP are based on the EM250, and not the TI CC2420. Utilities are requiring the security and standardization that the SEP provides. ...”
  - Bruce

EM250

- 12MHz XAP2b 16-bit microcontroller core
  - 128kB Flash and 5kB RAM
  - 128-bit AES hardware engine
  - <1uA sleep current w/ internal RC oscillator running
- Also a radio.

So to be clear.

- The argument is:
  - The CC2420 is vulnerable.
  - The EM250 doesn't expose keys by SPI.
  - Therefore, EM250 boards are secure.
- The argument is wrong.
  - Let's see why!

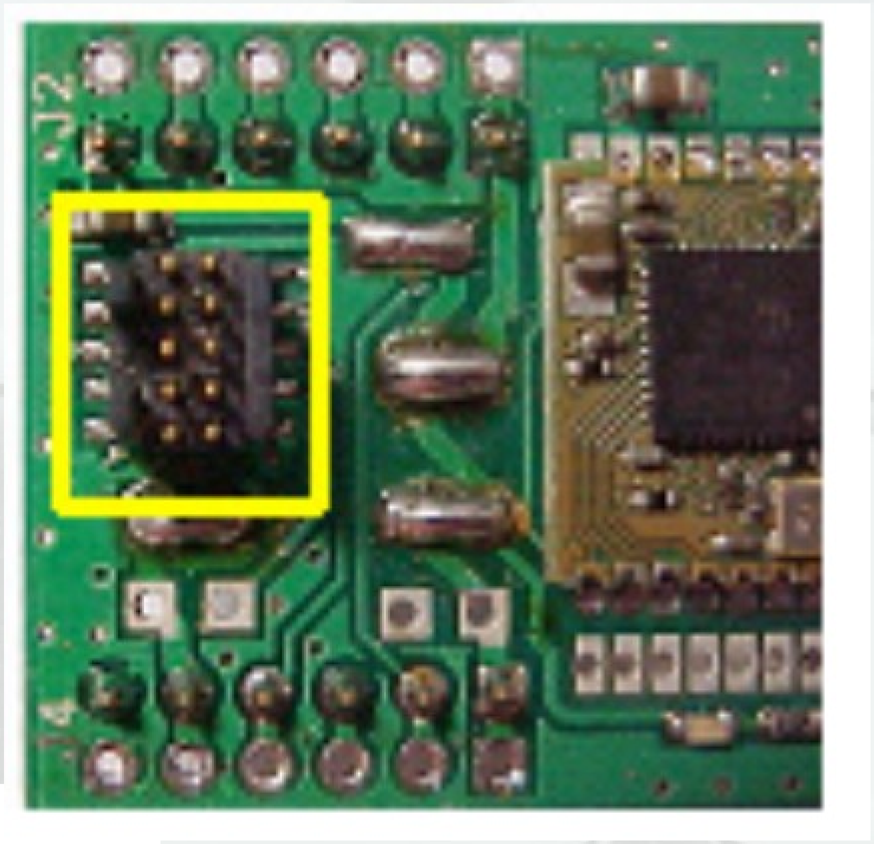
EM250 Chip

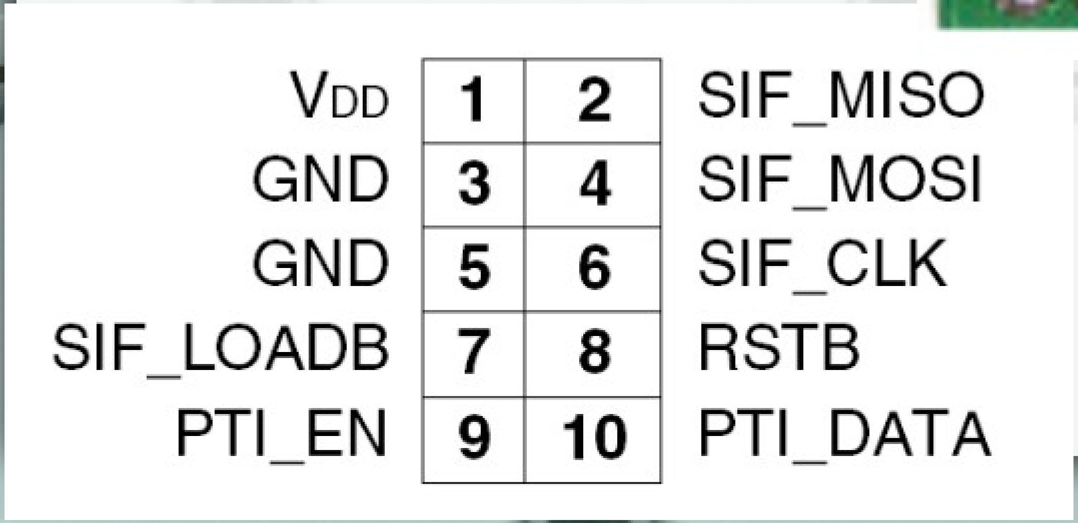
- 16 bit Harvard XAP2
  - 1999 design by Cambridge Consultants
- Insight® for Debugging
  - JTAG Variant

EM250 Programming

- OTA and by Serial Port
  - Bootloader of some sort.
  - Might be vulnerable. I haven't looked.
- Serial Port
  - Vulnerable to glitching, but don't bother.
- InSight®
  - Wide open.

InSight® Port





V_{DD}	1	2	SIF_MISO
GND	3	4	SIF_MOSI
GND	5	6	SIF_CLK
SIF_LOADB	7	8	RSTB
PTI_EN	9	10	PTI_DATA

from SPZB260ADP

Debug Port
ember
InSight Port
RCM Power Select
Ext – Off – Int

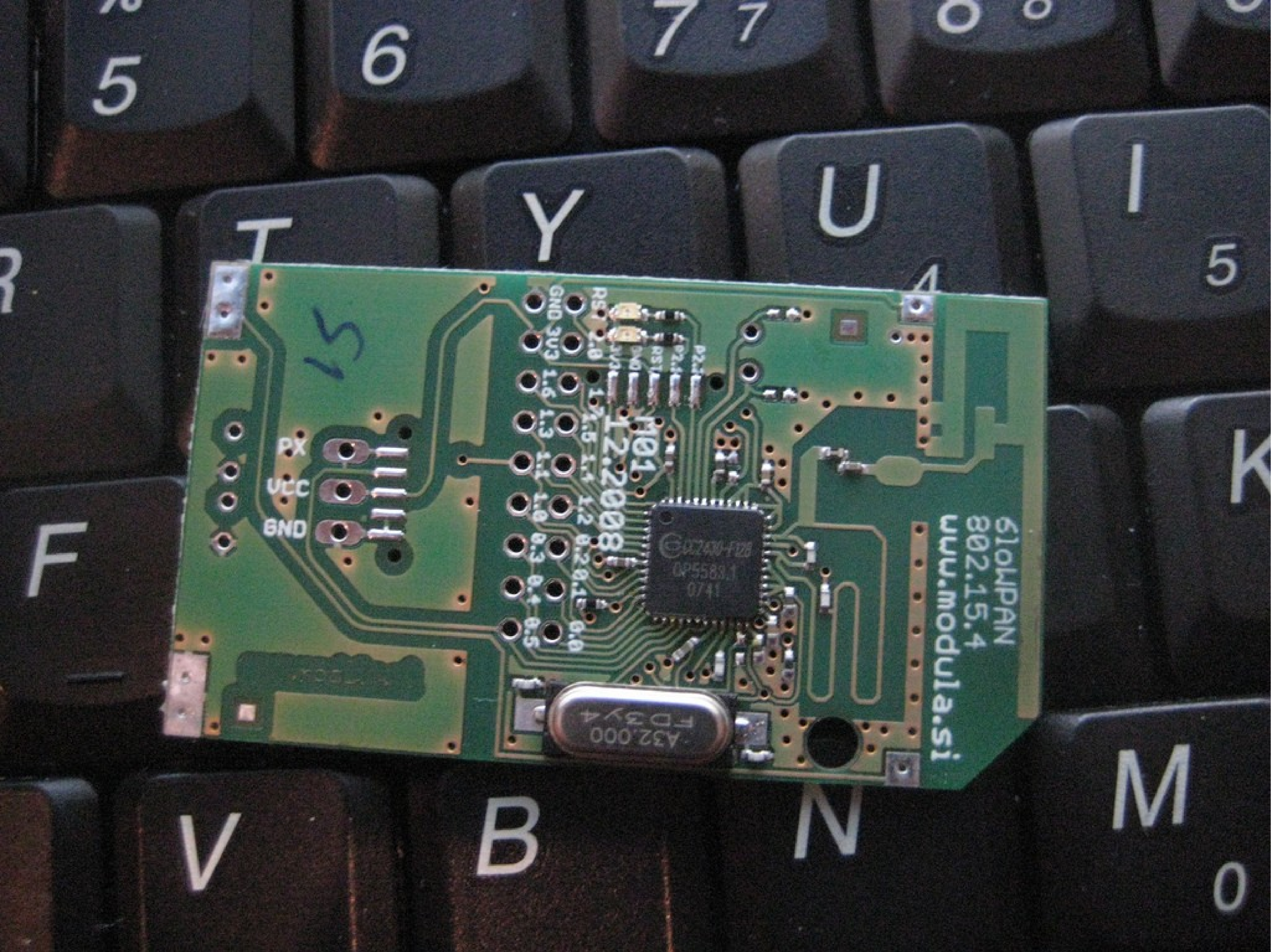
Insight® Cable

EM2xx Conclusions

- Insight®
  - Lacks a fuse.
  - Exploitable with Ember's own tools.
- Locally indefensible.

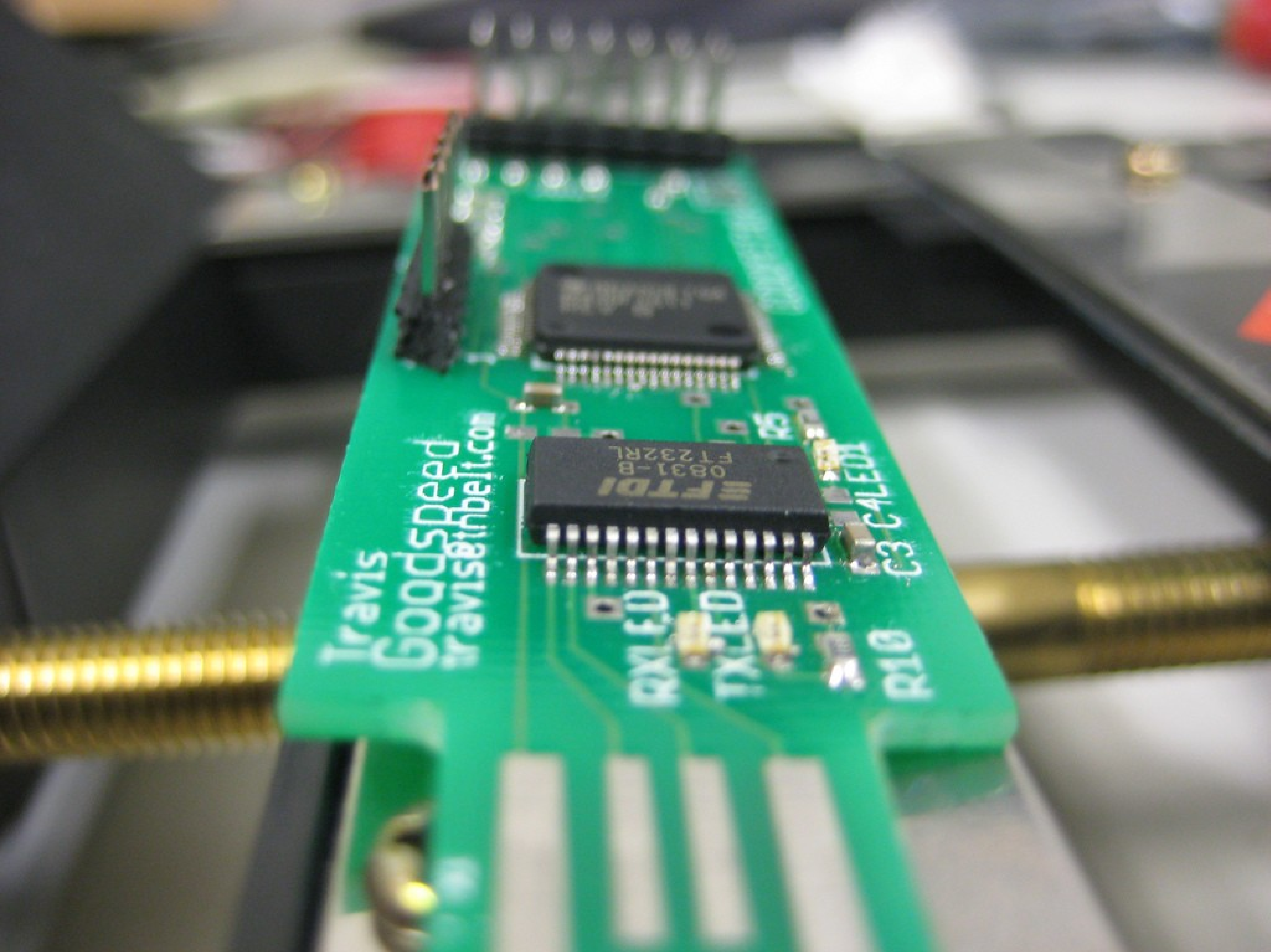
CC2430

- TI/Chipcon
- System on a Chip
  - 802.15.4 radio
  - 8051 µC
- Debugging
  - SPI-like
  - MOSI/MISO on a single pin.

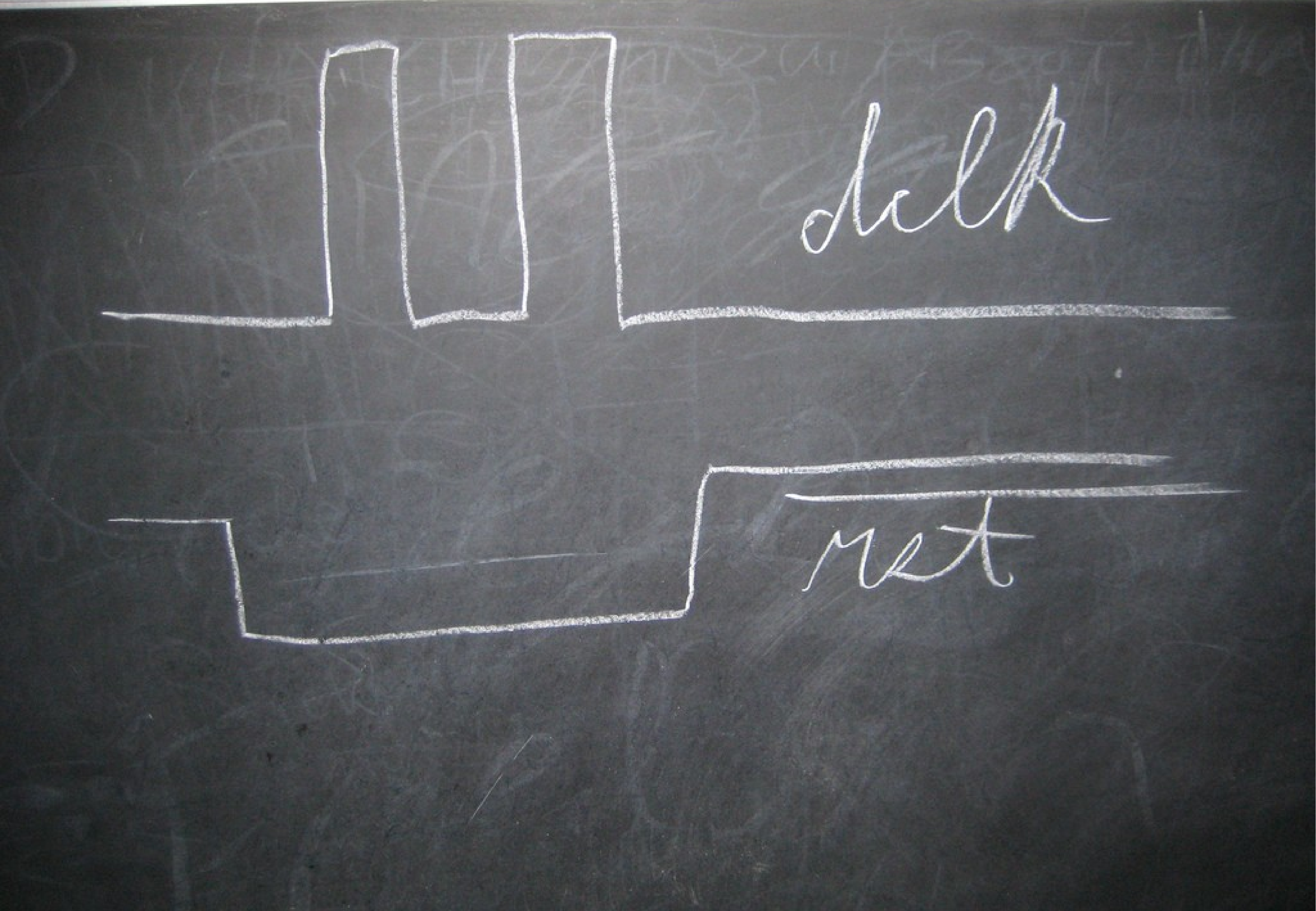

6loWPAN
802.15.4
www.modula.si
M01
12.2008
PX
VCC
GND
A32.000
15

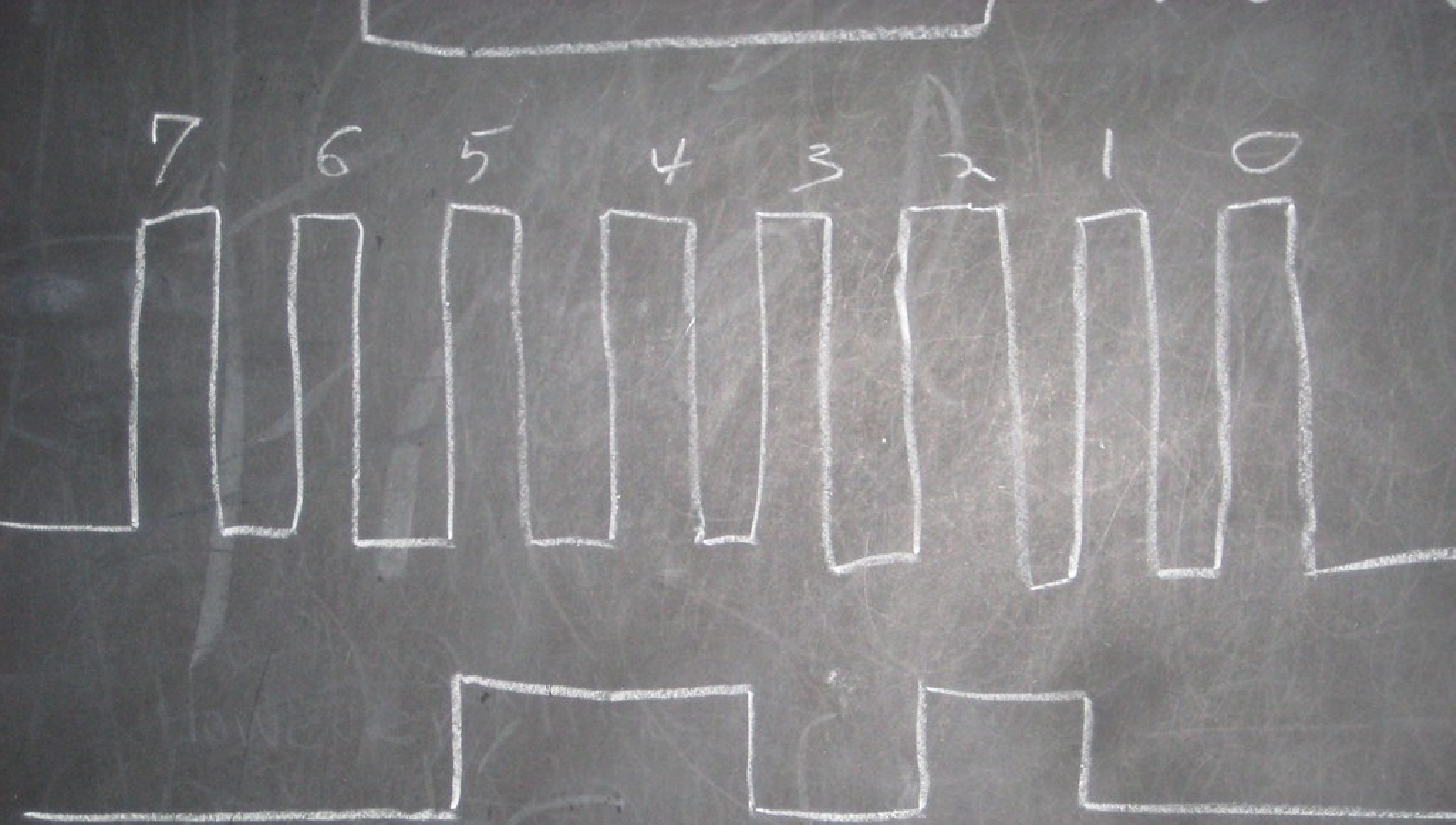
CC2430 Debugging

- Init Sequence
- Commands
  - CHIP_ERASE
  - GET_PC
  - DEBUG_INSTR
  - GET_CHIPID
- Reply


Travis
Goodspeed
travis@utbelt.com
FT232RL
0831-B
FTDI
C4LED1
C3
R5
RXLED
TXLED
R10

Entry Sequence





0x34

Chipcon Physical Layer

- Bits
  - MSBit first
  - Written on rising edge of clock.
  - Sampled on falling edge of clock.
- Direction
  - Master speaks first.
  - Slave replies.

Chipcon Debugging Protocol

- Command
  - 5b instruction
  - 1b R/!R
  - 2b Objects
- 0 to 3 object bytes
- 0 to 1 return bytes

Chipcon Lock Bit

- Unlocked
  - All verbs work.
- Locked
  - CHIP_ERASE
  - READ_STATUS
  - GET_CHIP_ID
- To unlock,
  - CHIP_ERASE

Chipcon CHIP_ERASE

- Erases all of Flash.
  - All firmware.
  - Debug Fuse too.
- None of RAM.

8051 Constant Sidebar

- 8051 is Modified Harvard Architecture
  - Data Memory
    - Non-executable.
    - Quickly read/written.
  - Code Memory
    - Executable.
    - Slowly read as data.
  - Incompatible pointers.

Brief Review

- Von Neumann
- Unified Memory
- Executable RAM

- Harvard
- Divided Memory
  - Code
  - Data
- Unexecutable RAM

8051 Constant Sidebar

- 8051 Compilers
  - All variables in Data memory,
    - unless explicitly told otherwise.
  - At initialization
    - Data is populated from Code.
- Therefore,
  - EVERY variable is in Data by default.
  - Keys are in Data memory.

Chipcon Exploitation

- GoodFET.CC
  - Erase
  - Write Data >keys.bin
- Key search
  - Joshua Wright's Killer Bee, TBR
  - 2 seconds for upper RAM
  - 4 seconds for all of RAM

Chipcon Defense

- Keep anything sensitive in Code memory.
  - See Chipcon DN200.
- const __code char foo[]=”Hello World!”
- printf(foo);
  - Won't work!
  - printf() expects a pointer to Data memory.

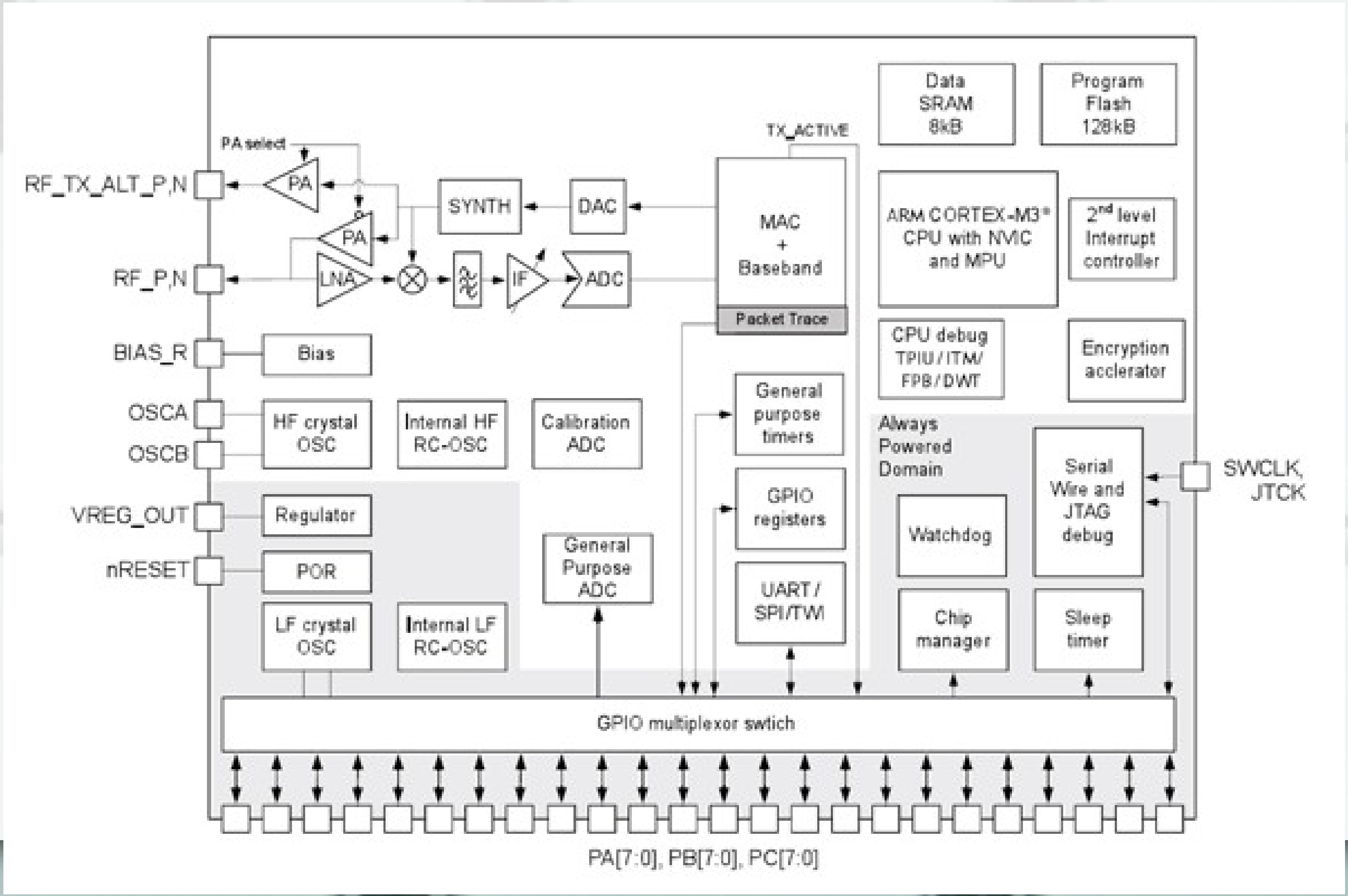
Chipcon Summary

- All current chips are vulnerable.
- Keys are exposed unless protected.
- Protection requires some recoding.

Third Generation Chips

- EM3xx
  - ARM Cortex M3 µC
  - JTAG Pin Fuse
- CC430
  - MSP430 µC
  - JTAG TAP Fuse
- Neither is yet available.

Third gen Chips: EM3xx



Third Gen Chips: CC430

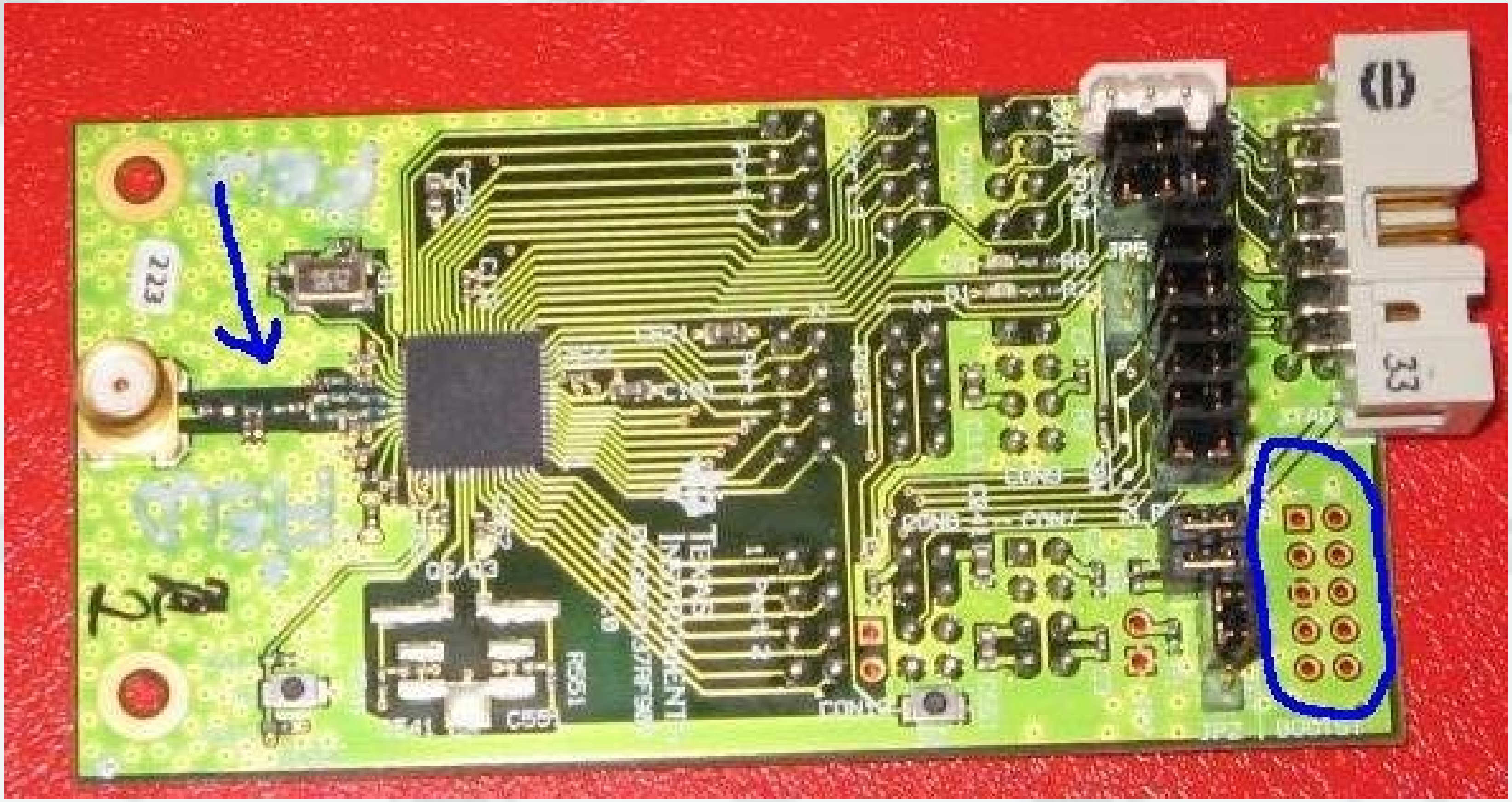


photo from TI E2E Blog

Part 1 Conclusions

- Zigbee chips aren't very secure.
- Next generation might be better.
  - Might not be better.
- Local security is hard.
  - Cryptography != Security

Part 2:
A 16-bit Rootkit

- IVT Proxying/Hooking
- Initial Foothold
- Blind Command Reception
- Efficient Command Frames
- Blind Function Calling

History

- 2007, I authored the first WSN exploit.
  - MSP430 infected by 802.15.4 packet
- 2008, I authored an MSP430 R.E. kit.
  - http://msp430static.sf.net/ in Perl/SQLite
- 2009, Mike Davis Smart Grid Worm
  - Catch his talk at 16h45.
  - Practical implementation, which mine ain't.

WSN Exploits in Brief

- Memory is precious
  - A few kilobytes of free memory.
  - 128 byte packets
- No operating system.
  - No system calls, function tables, etc.
  - Single statically-linked image.
- Code is in Flash, not RAM.

This Rootkit

- Generic Installation
  - Reasonably hardware agnostic.
  - Coexists with prior firmware.
- Efficient
  - Fits in available memory.
  - Reuses victim code where possible.
  - Memory/security tradeoff.

MSP430

- 16 bit RISC processor
  - Two 20 bit variants.
- Masked ROM Bootloader (BSL)
  - Flash ROM in recent variants.
- Chosen for a concrete example.
  - Similarities in AVR, PIC, MIPS, etc.

Rootkit Specifics

- How do you find a function?
  - No linking tables.
- How do you trap an incoming packet?
  - Radio drivers are inlined.
- How do you make the rootkit stealthy?
  - Would you make it stealthy?

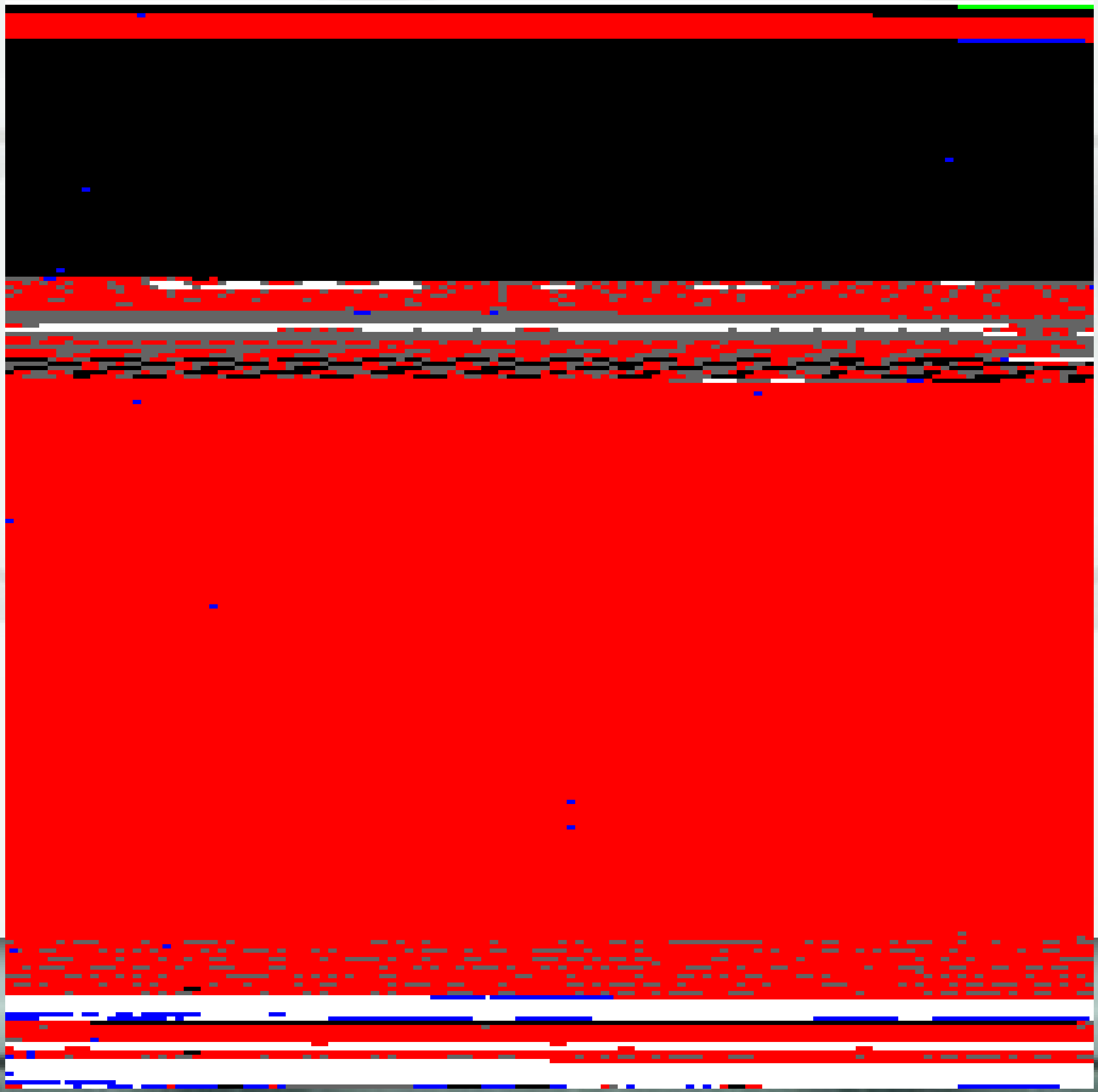
Locating a Function

- Fingerprints
  - Isolate functions, then iterate.
  - Checksum bytes.
  - Call function that matches bytes.
- Ports
  - IO ports are unique to hardware.
  - Called as literal indirects.

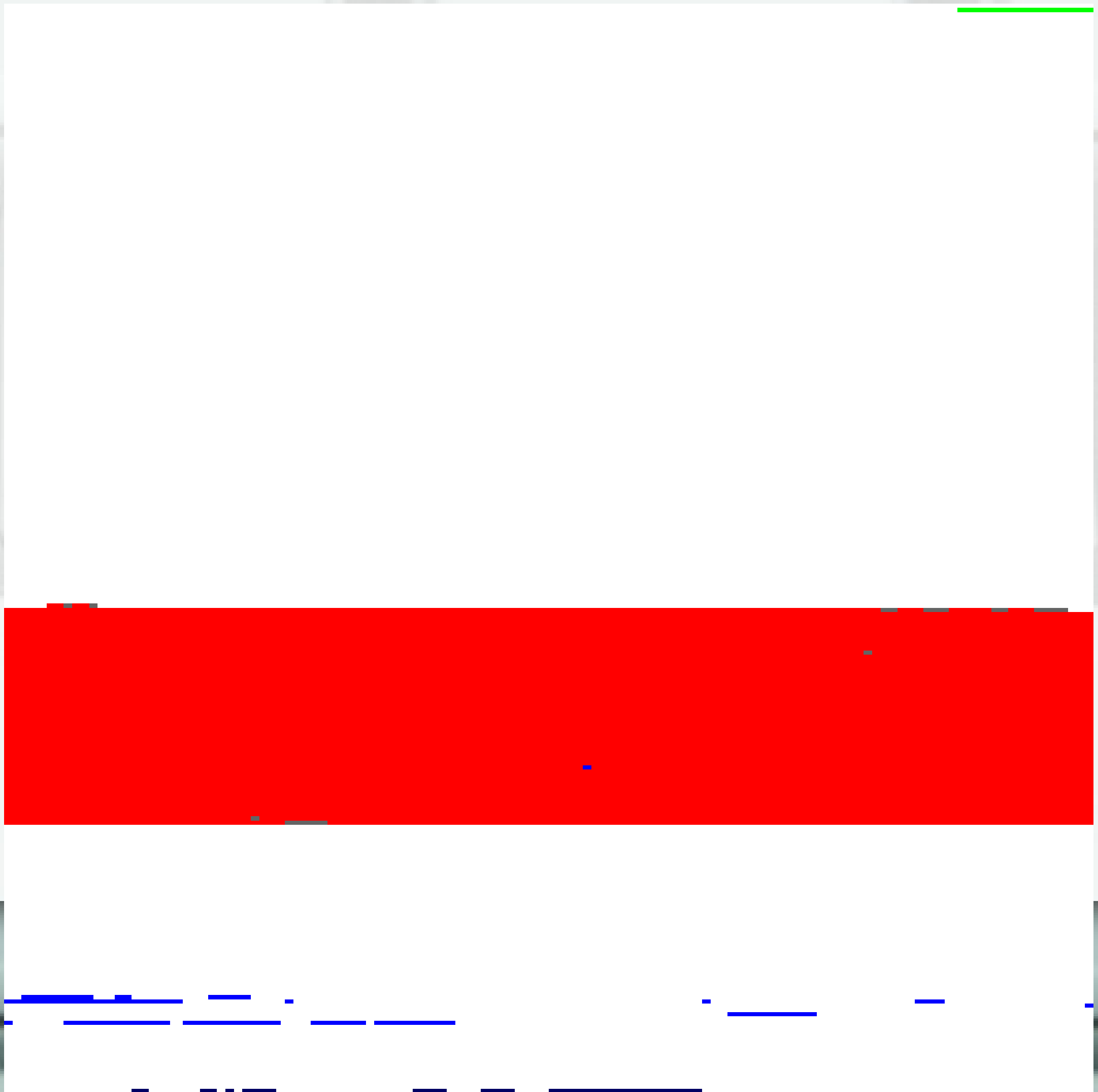
Interrupt Handling

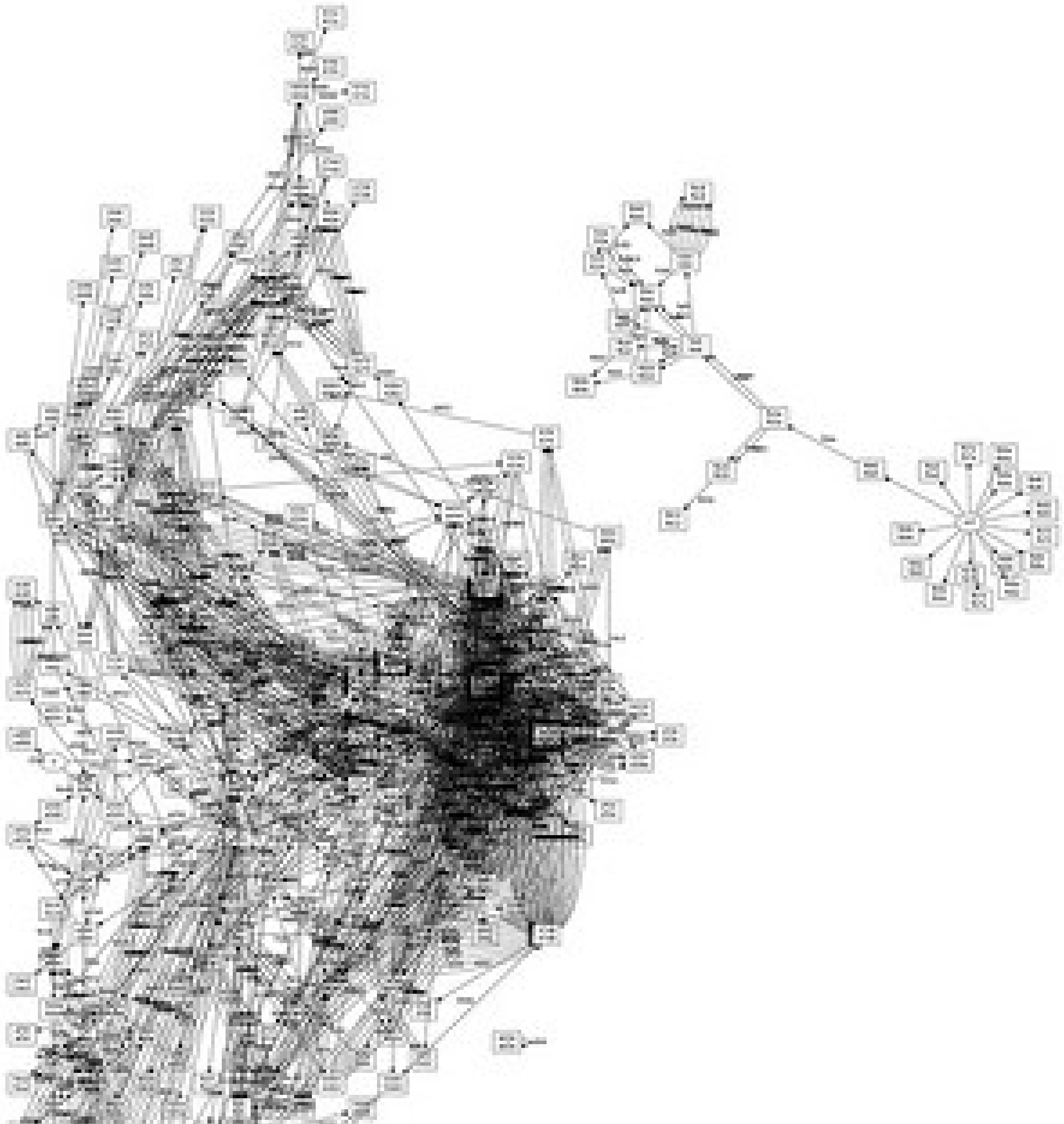
- Interrupt Vector Table
  - List of interrupt handler addresses.
  - At the top of memory in Flash.
- To proxy it,
  - Copy table to a lower address.
  - Handle each target.
  - Handler branches to original.

Interrupt Proxy



Unproxied





Interrupt Proxying

- Also used without malice.
- Drastically changes
    - Bootloader password.
    - Call Graph.
    - Memory usage.
    - Calling convention.
- Barely changes
    - Bytes.

Bootloader Password

- Hard to fake for masked BSL.
  - Entry sequence is in hardware.
  - Not maskable on classic MSP430.
- JTAG Fuse
  - If blown, access is restricted without pass.
  - If unblown, local attacker has access.

Call Graph

- Two applications,
  - Two disconnected graphs.
  - Child connections can be made,
    - CALL #0x4000
  - Parent connections are more difficult.
    - Clearing bits is easier than setting them.
    - Reflashing a segment.

Memory Usage

- Linker behavior
  - Flash is at the top of memory.
  - Code grows from starting address upward.
  - Each app starts at a segment boundary.

Calling Convention

- Hackers use GCC
  - r15, r14, r13, r12
- Others use IAR
  - r12, r14 in IAR 3
  - r12, r13, r14, r15 in IAR 4
- Other compilers
  - other conventions

Further Fingerprinting

- switch(){}
  - Table, word offset, or byte offset?
- mov #0xFFFF, r15
  - Constant generator or literal?
- Unused interrupts.
  - 0xFFFF, single handler, or many handlers?

Locating a Rootkit

- One app or two?
  - Memory map, register usage, gap.
- One compiler or two?
  - Calling convention consistency?
  - Assembler, switch{} consistency?

Two IVTs

- 0xFB78
- 0xFB7C
- 0xFB80
- 0xFB84
- 0xFB88
- 0xFB8C
- ...

- 0x403A, repeated
- 0x40B4
- 0x4068
- 0x43B8
- 0x40FA
- 0x4000

Once again,

- In IT,
  - Malware is common.
  - It's annoying.
  - Simple malware is detected, removed.
- In embedded systems,
  - Malware is rare.
  - No one looks for it.
  - Simple malware is undetected, sufficient.

For more information,

- TravisGoodspeed.blogspot.com
  - Compiler behavior survey.
  - MSP430static R.E. toolkit.
- GoodFET.sourceforge.net
  - Chipcon debugging.
  - Voltage glitching soon.

Defcon talks

- Locally Exploiting Wireless Sensors
  - Less theory, more practice.
- An Open JTAG Debugger
  - Mapping JTAG Registers
  - CC2430 Protocol
  - Voltage Glitching

Questions?