

0x0000

```
main:sudo
File Edit View Bookmarks Settings Help
Davids-iPhone-2.local., (Cache flush) A 169.254.202.124 (596)
21:59:40.574693 ARP, Request who-has 169.254.110.169 tell 0.0.0.0, length 46
21:59:40.766113 IP6 fe80::cd3f:7dcc:3f2a:d9b6:58124 > ff02::1:3.5355: UDP, length 24
21:59:40.766515 IP 169.254.217.182.51303 > 224.0.0.252.5355: UDP, length 24
21:59:40.769056 IP 169.254.85.161.137 > 169.254.255.255.137: NBT UDP PACKET(137): QUERY; REQUEST; BROADCAST
21:59:40.769572 IP 169.254.85.161.138 > 169.254.255.255.138: NBT UDP PACKET(138)
21:59:40.769916 IP6 fe80::f2b4:79ff:feb9:8bf.5353 > ff02::fb.5353: 0 [1n] [1au] ANY (QM)? Isaac.local. (80)
21:59:40.770321 IP6 fe80::f2b4:79ff:feb9:8bf.5353 > ff02::fb.5353: 0 [1n] [1au] ANY (QM)? Isaac.local. (80)
21:59:40.771293 IP6 fe80::a844:846f:6b64:53bd > ff02::1:ffa5:b075: ICMP6, neighbor solicitation, who has fe80::7ae7:d1ff:fea5:b075, length 32
21:59:40.771678 IP 169.254.88.88.137 > 169.254.255.255.137: NBT UDP PACKET(137): QUERY; REQUEST; BROADCAST
21:59:40.771893 STP 802.1d, Config, Flags [none], bridge-id 5014.a0:cf:5b:d6:03:80.8024, length 42
21:59:40.772198 ARP, Request who-has 169.254.244.249 tell 0.0.0.0, length 46
21:59:40.772533 IP6 fe80::cd3f:7dcc:3f2a:d9b6:58124 > ff02::1:3.5355: UDP, length 24
21:59:40.776037 IP 169.254.217.182.51303 > 224.0.0.252.5355: UDP, length 24
21:59:40.776602 IP 169.254.135.98.5353 > 224.0.0.251.5353: 0* [0q] 3/0/2 TXT "model=MacBookPro5,4", (Cache flush) PTR Tobias-Selliers-MacBook-Pro.local., (Cache flush) A 169.254.135.98 (218)
21:59:40.776925 IP6 fe80::6c50:a153:893a:f5.51307 > ff02::1:3.5355: UDP, length 34
21:59:40.777331 ARP, Request who-has 192.168.0.1 tell 192.168.0.162, length 46
21:59:40.778655 IP 169.254.0.245.54692 > 224.0.0.252.5355: UDP, length 34
21:59:40.779089 IP 169.254.174.214.137 > 169.254.255.255.137: NBT UDP PACKET(137): QUERY; REQUEST; BROADCAST
21:59:40.779391 IP 192.168.0.130.137 > 192.168.0.255.137: NBT UDP PACKET(137): QUERY; REQUEST; BROADCAST
21:59:40.779829 IP6 fe80::f2b4:79ff:feb9:8bf.5353 > ff02::fb.5353: 0 [3q] [1au] PTR (QM)? _ubd_tcp.Local. A (QM)? astec-exch.astec.local. AAAA (QM)? astec-exch.astec.local. (85)
21:59:40.780193 IP 169.254.138.142.137 > 169.254.255.255.137: NBT UDP PACKET(137): QUERY; REQUEST; BROADCAST
21:59:40.780597 IP6 fe80::f2b4:79ff:feb9:8bf.5353 > ff02::fb.5353: 0 [3q] [1au] PTR (QM)? _ubd_tcp.Local. A (QM)? astec-exch.astec.local. AAAA (QM)? astec-exch.astec.local. (85)
21:59:40.782948 ARP, Request who-has 192.168.0.1 tell 192.168.0.130, length 46
21:59:44.003264 IP 0.0.0.0.68 > 255.255.255.255.67: BOOTP/DHCP, Request from 00:23:15:af:30:d0, length 300
21:59:44.657120 IP 169.254.136.101.137 > 169.254.255.255.137: NBT UDP PACKET(137): QUERY; REQUEST; BROADCAST
21:59:44.658042 ARP, Request who-has 192.168.0.1 tell 192.168.0.162, length 46
21:59:44.660152 ARP, Request who-has 169.254.126.115 tell 169.254.126.115, length 46
21:59:44.662331 IP6 fe80::3651:c9ff:fecf:5ec7 > ff02::2: ICMP6, router solicitation, length 16
21:59:44.662622 ARP, Request who-has 192.168.10.1 tell 192.168.10.180, length 46
21:59:44.662926 ARP, Request who-has 192.168.0.1 tell 192.168.0.130, length 46
21:59:44.663847 IP 169.254.217.182.137 > 169.254.255.255.137: NBT UDP PACKET(137): QUERY; REQUEST; BROADCAST
21:59:44.664353 IP6 fe80::f18e:5b34:ecal:1aec.53648 > ff02::c.1900: UDP, length 146
21:59:44.664770 IP6 fe80::f18e:5b34:ecal:1aec > ff02::1:ff95:8ea7: ICMP6, neighbor solicitation, who has fe80::55b3:cb9c:3395:8ea7, length 32
21:59:44.665258 IP6 fe80::6c50:a153:893a:f5.51103 > ff02::1:3.5355: UDP, length 37
21:59:44.669822 IP 169.254.0.245.57179 > 224.0.0.252.5355: UDP, length 37
21:59:44.670154 IP6 fe80::6c50:a153:893a:f5.63274 > ff02::1:3.5355: UDP, length 37
21:59:44.674676 IP 169.254.0.245.65360 > 224.0.0.252.5355: UDP, length 37
21:59:44.674961 ARP, Request who-has 192.168.0.1 tell 192.168.0.227, length 46
21:59:44.675335 IP 169.254.244.249.5353 > 224.0.0.251.5353: 0 [2q] [2n] [1au] ANY (QU)? iPad-69.local. ANY (QU)? iPad-69.local. (104)
21:59:44.677318 IP6 fe80::72de:e2ff:fea4:92c2.5353 > ff02::fb.5353: 0 [2q] [2n] [1au] ANY (QU)? iPad-69.local. ANY (QU)? iPad-69.local. (104)
21:59:44.677694 IP6 fe80::72de:e2ff:fea4:92c2.5353 > ff02::fb.5353: 0 [2q] [2n] [1au] ANY (QU)? iPad-69.local. ANY (QU)? iPad-69.local. (104)
21:59:44.677934 IP6 fe80::cd3f:7dcc:3f2a:d9b6:53944 > ff02::1:3.5355: UDP, length 22
21:59:44.678290 IP 169.254.217.182.62839 > 224.0.0.252.5355: UDP, length 22
21:59:44.678706 IP6 fe80::804f:a765:d83c:99a0.60905 > ff02::1:3.5355: UDP, length 22
21:59:44.679850 IP 169.254.153.160.55283 > 224.0.0.252.5355: UDP, length 22
21:59:44.680157 IP6 fe80::804f:a765:d83c:99a0.58307 > ff02::1:3.5355: UDP, length 22
21:59:44.680576 IP 169.254.153.160.54881 > 224.0.0.252.5355: UDP, length 22
21:59:44.680985 IP6 fe80::a52a:6f6:8e99:172f.58723 > ff02::1:3.5355: UDP, length 27
21:59:44.681394 IP 169.254.136.101.137 > 169.254.255.255.137: NBT UDP PACKET(137): QUERY; REQUEST; BROADCAST
21:59:44.681793 IP 169.254.23.47.52166 > 224.0.0.252.5355: UDP, length 27
21:59:44.682157 IP6 fe80::a667:6ff:fe8a:ffc9 > ff02::1: ICMP6, neighbor advertisement, tgt is fe80::a667:6ff:fe8a:ffc9, length 32
21:59:44.682583 IP6 fe80::a667:6ff:fe8a:ffc9 > ff02::2: ICMP6, router solicitation, length 16
21:59:44.683160 IP 0.0.0.0.68 > 255.255.255.255.67: BOOTP/DHCP, Request from 58:1f:aa:6e:a2:0d, length 300
21:59:44.683498 ARP, Request who-has 169.254.135.98 tell 169.254.153.160, length 46
21:59:44.683793 IP 169.254.16.139.137 > 169.254.255.255.137: NBT UDP PACKET(137): QUERY; REQUEST; BROADCAST
21:59:44.865628 IP 169.254.110.169.5353 > 224.0.0.251.5353: 0 [2q] [2n] [1au] ANY (QM)? iPhone-2.local. ANY (QM)? iPhone-2.local. (105)
```

“We are not as strong as we think we are”

• Rich Mullins

<GHz or bust!

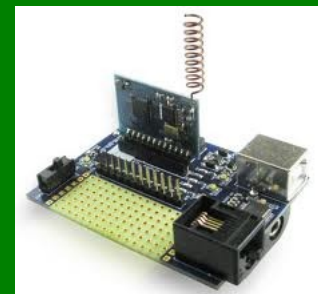
leveraging the power of the
chipcon 1111
(and RFCAT)

0x1000 – intro to <GHz

- FCC Rules(title 47) parts 15 and 18 allocate and govern parts of the RF spectrum for unlicensed ISM in the US (US adaptation of the ITU-R 5.138, 5.150, and 5.280 rules)
 - Industrial – power grid stuff and more!
 - Science -
 - Medical – insulin pumps and the like
- US ISM bands:
 - 300 :
 - 433 : 433.050 – 434.790 MHz
 - 915 : 902.000 – 928.000 MHz
- Popular European ISM band:
 - 868 : 863.000 – 870.000 MHz
- Other ISM includes 2.4 GHz and 5.8 GHz
 - cc2531.... hmmm... maybe another toy/talk?

0x1010 – what is <GHz? what plays there?

- Industry, Science, Medical bands, US and EU
- Cell phones
- Cordless Phones
- Personal Two-Way Radios
- Car Remotes
- Pink IM-ME Girl Toys!
- TI Chronos Watches
- Medical Devices (particularly 401-402MHz, 402-405MHz, 405-406MHz)
- Power Meters
- custom-made devices
- Old TV Broadcast
- much, much more...



- <INSERT PICTURES OF STUFF>

0x1020 – how do we play with it?



- cc1110/cc1111 do 300-348MHz, 391-464MHz, 782-928MHz
 - and more...
- RFCAT uses the CC111x on some common dongles
 - Chronos dongle (sold with every TI Chronos watch)
 - “Don's Dongles”, aka TI CC1111EMK
 - IMME (currently limited to sniffer/detection firmware)
- but there are some catches
 - rf comms configuration?
 - channel hopping sequence?
 - bluetooth and DSSS? (not hap'nin)



0x1030 – why do i care!?

- the inner rf geek in all of us
- your security research may require that you consider comms with a wireless device
- your organization may **have** 900MHz devices that should be protected!

0x2000 - intro to the cc1111 radio (the cc1100 core)

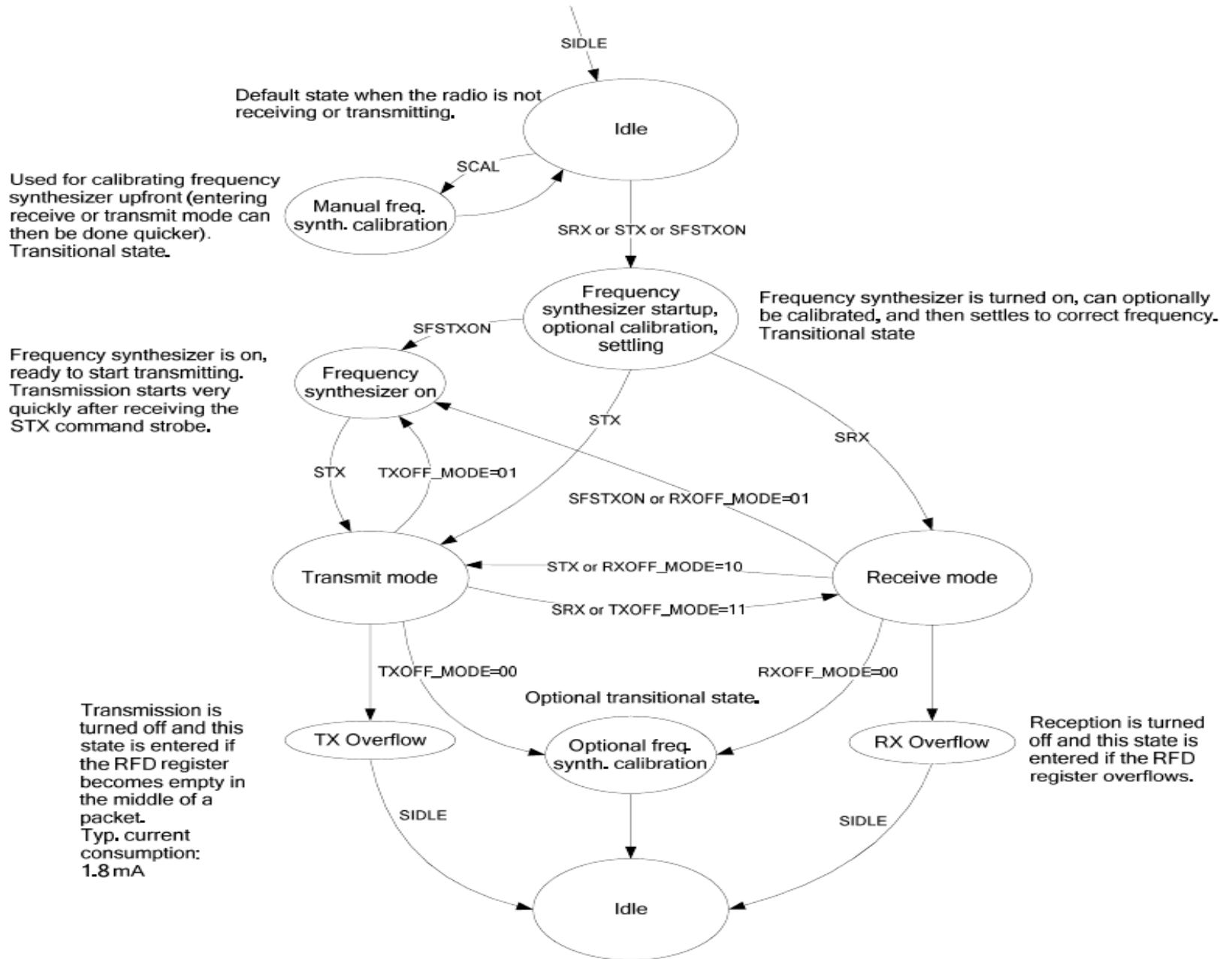
- mcu
- radio state engine
- radio configuration
- usb
- timers
- dma

0x2010 – cc1111 mcu

- modified 8051 core
 - 8-bit mcu
 - single-tick instructions
 - 256 bytes of iram
 - 4kb of xram
 - XDATA includes all code, iram, xram
 - execution happens anywhere :)
- register access to radio, dma, crypto, usb, timers, adc
- registers are simply memory locations in the XDATA address space

0x2020 – cc1111 radio state engine

- IDLE
- CAL
- FSTXON
- RX
- TX



0x2030 – cc1111 radio configuration

- configuring the radio is done through updating a set of 1-byte registers in varying bit-size fields
 - MDMCFG4 – MDMCFG0 – modem control
 - PKTCTRL1, PKTCTRL0 – packet control
 - FSCTRL1, FSCTRL0 – frequency synth control
 - FREQ2, FREQ1, FREQ0 – front end control
 - MCSM1, MCSM0 – radio state machine
 - SYNC1, SYNC0 – SYNC word, or the SFD
 - CHANNR, ADDR – channel and address
 - AGCCTRL2, AGCCTRL1, AGCCTRL0

0x2040 - Smart RF Studio (ftw)

The screenshot displays the Smart RF Studio software interface, which is used for configuring and controlling a CC1111 device. The main window is titled "CC1111 - Device Control Panel (offline)" and features a menu bar with "File", "Settings", "View", "Evaluation Board", and "Help".

The interface is divided into several sections:

- Typical settings:** A list of pre-configured settings for various data rates and modulation schemes, such as "Data rate: 1.2 kBaud, Dev.: 5.1 kHz, Mod.: GFSK, RX BW: 63 kHz, Optimized for sensitivity".
- RF Parameters:** A section for configuring the device's RF parameters, including:
 - Base frequency: 868.299683 MHz
 - Channel number: 0
 - Channel spacing: 199.951172 kHz
 - Carrier frequency: 868.299683 MHz
 - Xtal frequency: 48.000000 MHz
 - Data rate: 1.19877 kBaud
 - RX filter BW: 62.500000 kHz
 - Manchester enable:
 - Modulation format: GFSK
 - Deviation: 5.126953 kHz
 - TX power: 0 dBm
 - PA ramping:
- Operation Mode:** Buttons for "Continuous TX", "Continuous RX", "Packet TX", "Packet RX", and "RF Device Commands".
- Packet Configuration:** Fields for "Packet payload size" (30) and "Packet count" (100). A "Random" radio button is selected, and a hex payload is shown: "47 de b3 12 4d c8 43 bb 8b a6 1f 03 5a 7d 09 38 25 1f 5d d4 cb fc 96 f5 45 3b 13 0d 89 0a".
- TX/RX Status:** Visual indicators for TX and RX activity, along with status text: "Sent packets: 0", "Frequency: 868.299683 MHz", and "Output power: 0 dBm".
- Register View:** A separate window titled "CC1111 - Register View (offline)" showing a list of registers and their values in hexadecimal. The registers listed include IOCFG2, IOCFG1, IOCFG0, SYNC1, SYNC0, PKTLEN, PKTCTRL1, PKTCTRL0, ADDR, CHANNR, FSCTRL1, FSCTRL0, FREQ2, FREQ1, FREQ0, MDMCFG4, MDMCFG3, MDMCFG2, MDMCFG1, MDMCFG0, DEVIATN, MCSM2, MCSM1, MCSM0, FOCCFG, BSCFG, AGCCTRL2, AGCCTRL1, AGCCTRL0, FREND1, FREND0, FSCAL3, and FSCAL2.

0x2050 – cc1111 radio notes

- Data Rate, Bandwidth, and Intermediate Frequency and Freq-Deviation depend on each other
- put the radio in IDLE state before configuring
- put the radio in IDLE state before configuring
- put the radio in IDLE state before configuring
- STROBE (SIDLE, STX, SRX, SCAL...)
 - then wait for the MARCSTATE == MARC_STATE_whatever
- CCA impacts entering TX state from RX
 - but not from IDLE state

0x2100 – usb

- usb is a world unto itself, with a massive standard and substandards (gg: usb-in-a-nutshell)
- cc1111's usb controller is accessed using:
 - registers for config/control of usb
 - registers indicating usb events that occur
 - endpoint-specific FIFO buffers
 - messages go there before sending to host
 - messages arrive there from host
 - usb “descriptors” as necessary by spec
 - host uses these to query the device
- our firmware provides all this and more

0x2110 – usb for devs

- application.c provides the template for new apps
 - copy it and make
- cc1111usb.c provides usb descriptors and framework
- txdata(buffer, length) to send data IN to host
- registerCbEP5OUT() to register a callback function to handle data OUT from host
 - data is in ep5iobuf[]
- follow the example, luke!

0x3000 – what radio things do we want to know!?

- frequencies
- modulation (2FSK/GFSK, MSK, ASK/OOK, other)
- intermediate frequency (IF)
- baud rate
- channel width/spacing/hopping?
- bandwidth filter
- sync words / bit-sync
- variable length/fixed length packets
- crc
- data whitening?
- encoding (manchester, fec, enc, etc...)

0x3010 – frequencies

- 315MHz – car fobs
- 433MHz – medical devices, EU loves this range
- 868MHz – EU loves this range too
- 915MHz – NA stuff of all sorts (power meters, insulin pumps, industrial plant equipment, industrial backhauls)
- 2.4GHz – 802.11/wifi, 802.15.4/zigbee/6lowpan, bluetooth
- 5.8GHz – cordless phones
- FREQ2, FREQ1, FREQ0

0x3020 – modulations

- 2FSK/GFSK – Frequency Shift Key

- (digital FM)
- cordless phones (DECT/CT2)

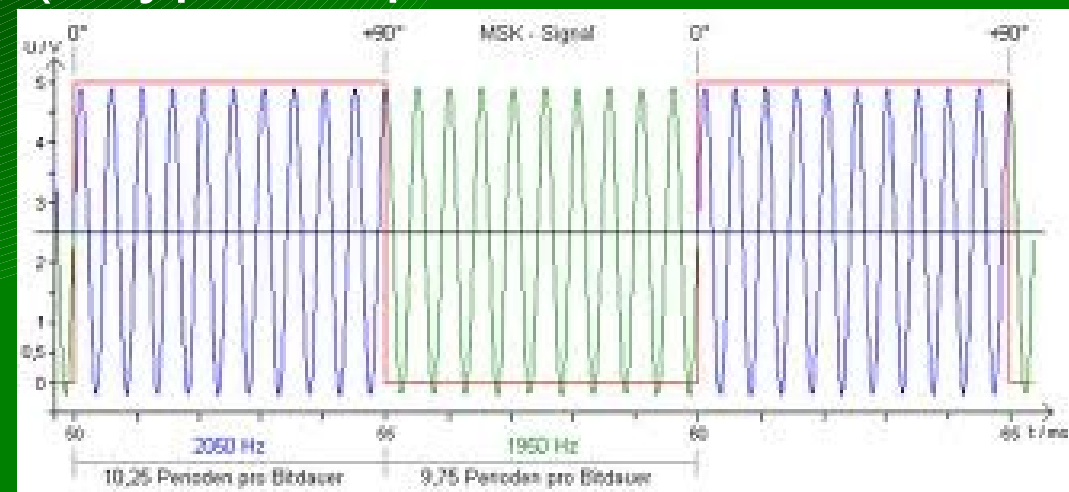
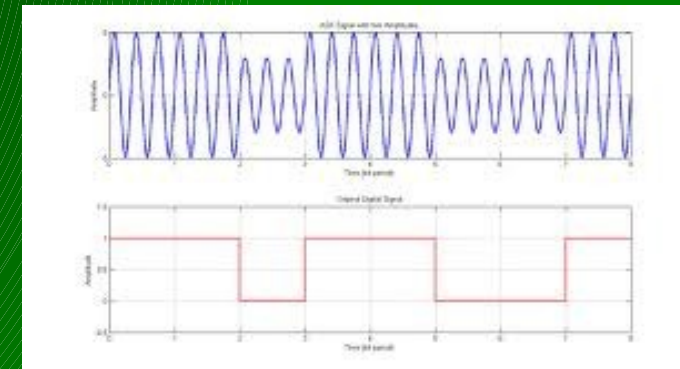
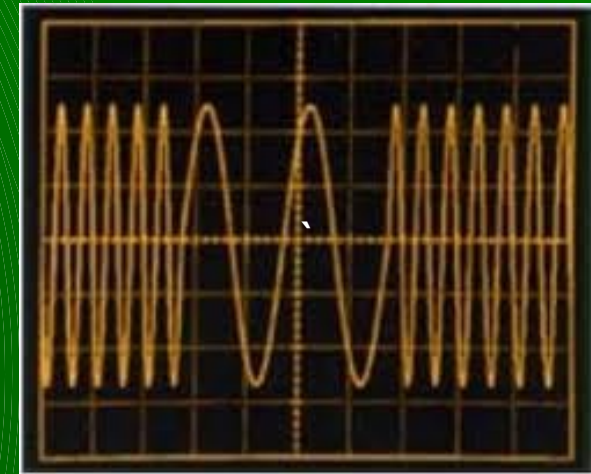
- ASK/OOK – Amplitude Shift Key

- (digital AM)
- morse-code, car-remotes, etc...

- MSK – Minimal Shift Key (a type of quadrature shift modulation like QPSK)

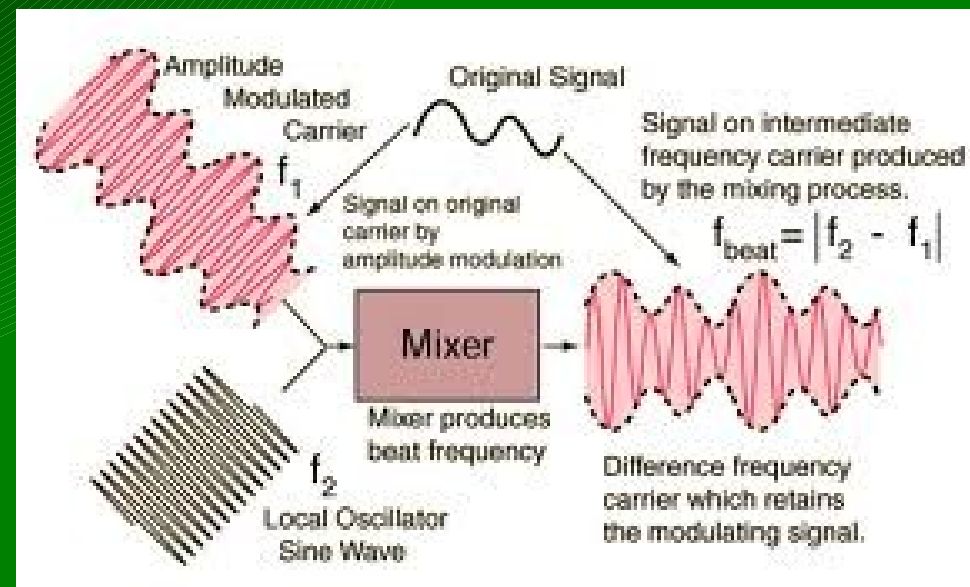
- GSM

- MDMCFG2, DEVIATN



0x3030 – intermediate frequency

- mix the RF and LO frequencies to create an IF
 - improves signal selectivity
 - tune different frequencies to an IF that can be manipulated easily
 - cheaper/simpler components
- cc1111 supports a wide range of 31 different IF options:
 - 23437 hz apart, from 0 – 726.5 khz
- Smart RF Studio recommends:
 - 140 khz up to 38.4 kbaud
 - 187.5 khz at 38.4 kbaud
 - 281 khz at 250 kbaud
 - 351.5khz at 500 kbaud
- FSCTRL1

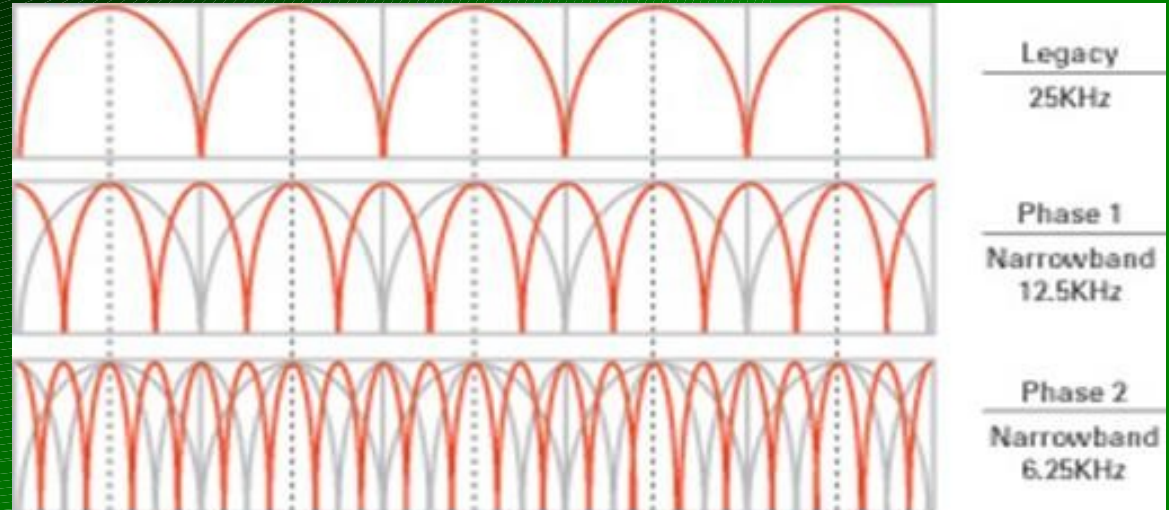


0x3040 – data rate (baud)

- much like your modems or old
- the frequency of bits
 - some can overlap and get garbage!
 - garbage can be good...
- baud has significant impact on IF, Deviation and Channel BW
- seeing use of 2400, 19200, 38400, 250000
- MDMCFG3 / 4

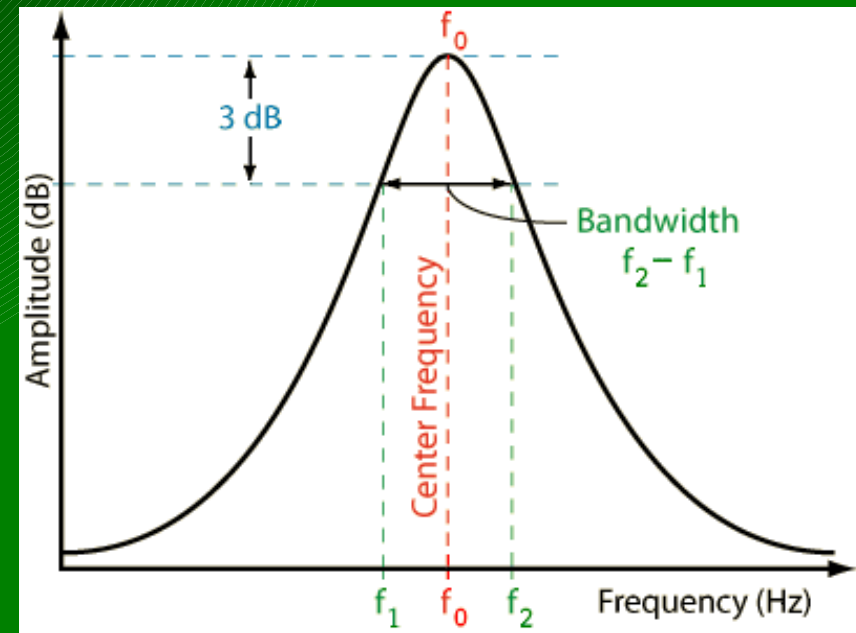
0x3050 – channel width / spacing

- simplifying frequency hopping / channelized systems
- $\text{real freq} = \text{base freq} + (\text{CHANNR} * \text{width})$
- MDMCFG0 / 1



0x3060 – bandwidth filter

- programmable receive filter
- provides for flexible channel sizing/spacing
- total signal bw = signal bandwidth + (2*variance)
- total signal bw **wants** to be less than 80% bw filter!
- MDMCFG4



0x3070 – preamble / sync words

- identify when real messages are being received!
- starts out with a preamble (1 0 1 0 1 0 1 0...)
- then a sync word (programmable bytes)
 - marking the end of the preamble
 - SFD – start of frame delimiter
- configurable to:
 - nothing (just send received crap)
 - carrier detect (if the RSSI value indicates a message)
 - 15 or 16 bits of the SYNC WORD identified
 - 30 out of 32 bits of double-SYNC WORD
- SYNC1, SYNC0, MDMCFG2

0x3080 – variable / fixed-length packets

- packets can be fixed length or variable length
- variable length assumes first bytes is the length byte
- both modes use the PKTLEN register:
 - Fixed: the length
 - Variable: MAX length
- PKTCTRL0

0x3090 – CRC – duh, but not

- crc16 check on both TX and RX
- uses the internal CRC (part of the RNG) seeded by 0xffff
- DATA_ERROR triggers when CRC is enabled and fails
- PKTCTRL0

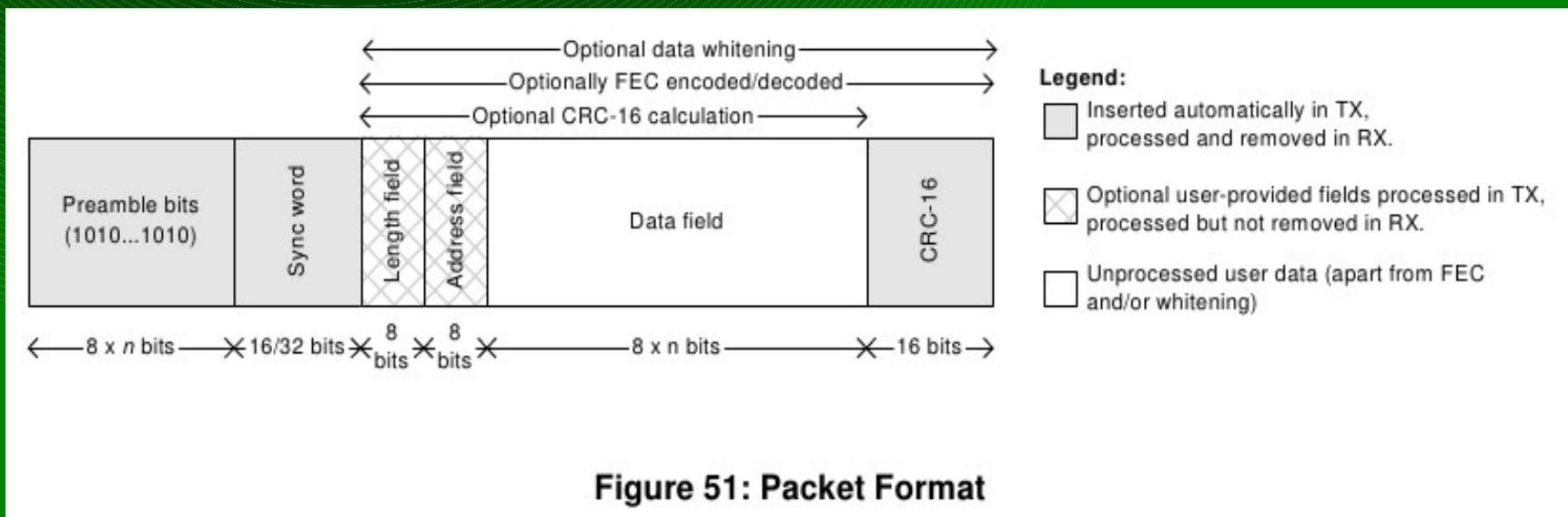


Figure 51: Packet Format

0x30a0 – data whitening – 9 bits of pain

- ideal radio data looks like random data
- real world data can contain long sequences of 0 or 1
- data to be transmitted is first XOR'd with a 9-bit sequence
 - sequence repeated as many times as necessary to match the data

- PKTCTRL0

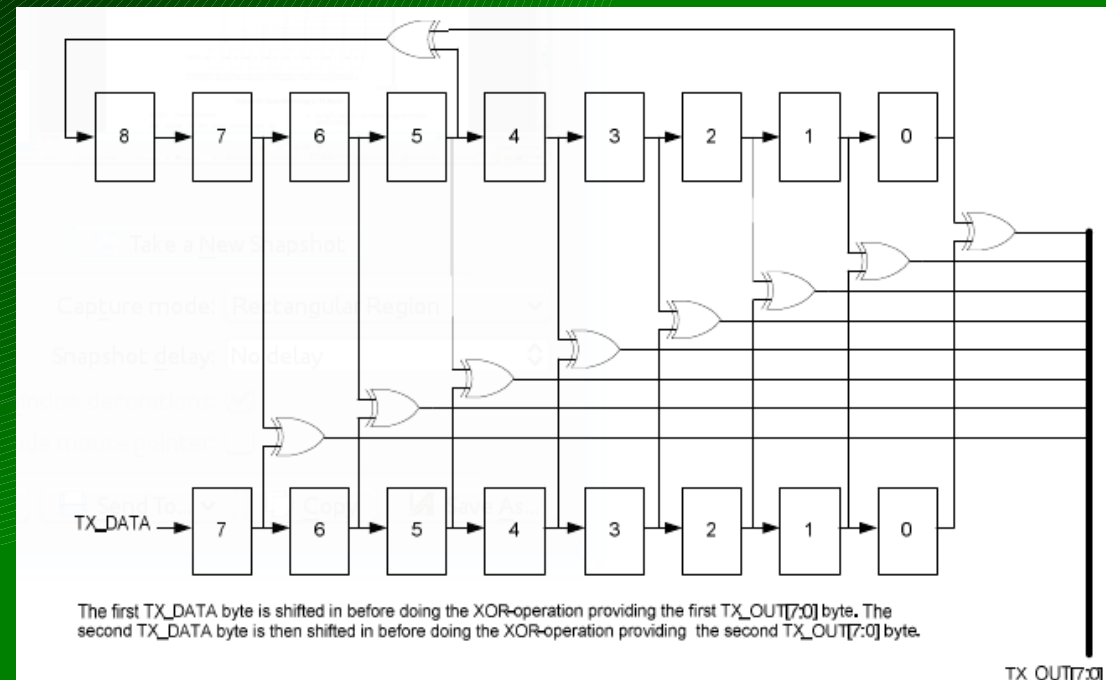
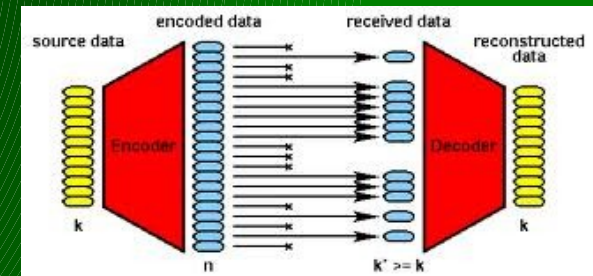
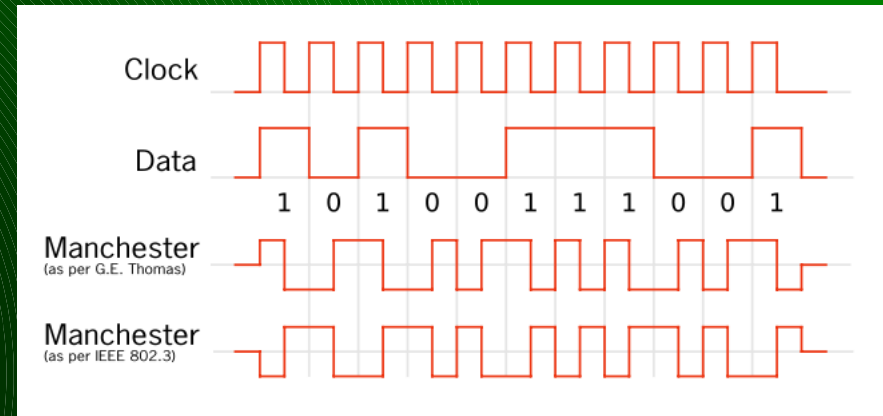


Figure 50: Data Whitening in TX Mode

0x30b0 – encoding

- manchester
 - MDMCFG2
- forward error correction
 - convolutional
 - MDMCFG1
 - reed-solomon (not supported)
- encryption - AES in chip



AES Crib Sheet (Handy for memorizing)

Plaintext in 4x4 grid

Initial Round

General Math

- 1.1B = AES Polynomial: $x^4 + x^3 + x^2 + x + 1$
- Fast Multiply
- $x^2 \cdot x^3 = x^5 = x^1 \pmod{1.1B}$
- $x \cdot a(x) = (a \ll 1) \oplus 1B$ ($a_2 = 1$): 00
- $\log(x \cdot y) = \log(x) + \log(y)$
- Use $(x+1) = 03$ for log base

Fast Multiply

Final Round

Key Expansion: Round Constants

First Column: 01 02 04 08

Round Key

Other Columns:

S	B	E	9
0	1	2	3
4	5	6	7
8	9	A	B
C	D	E	F
10	11	12	13
14	15	16	17
18	19	1A	1B
1C	1D	1E	1F

Mix Columns:

2	13	3	12
10	9	15	4
14	5	6	11
8	7	16	17

Inverse Mix

E	B	D	9
4	A	5	14
15	10	11	13
8	7	16	17

Shift Rows Row Shift

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Intermediate Rounds

Round #	Key
9	128
11	192
13	256

Ciphertext

?	?	?	?
?	?	?	?
?	?	?	?
?	?	?	?

Prev Col ⊕ Col from Previous round key

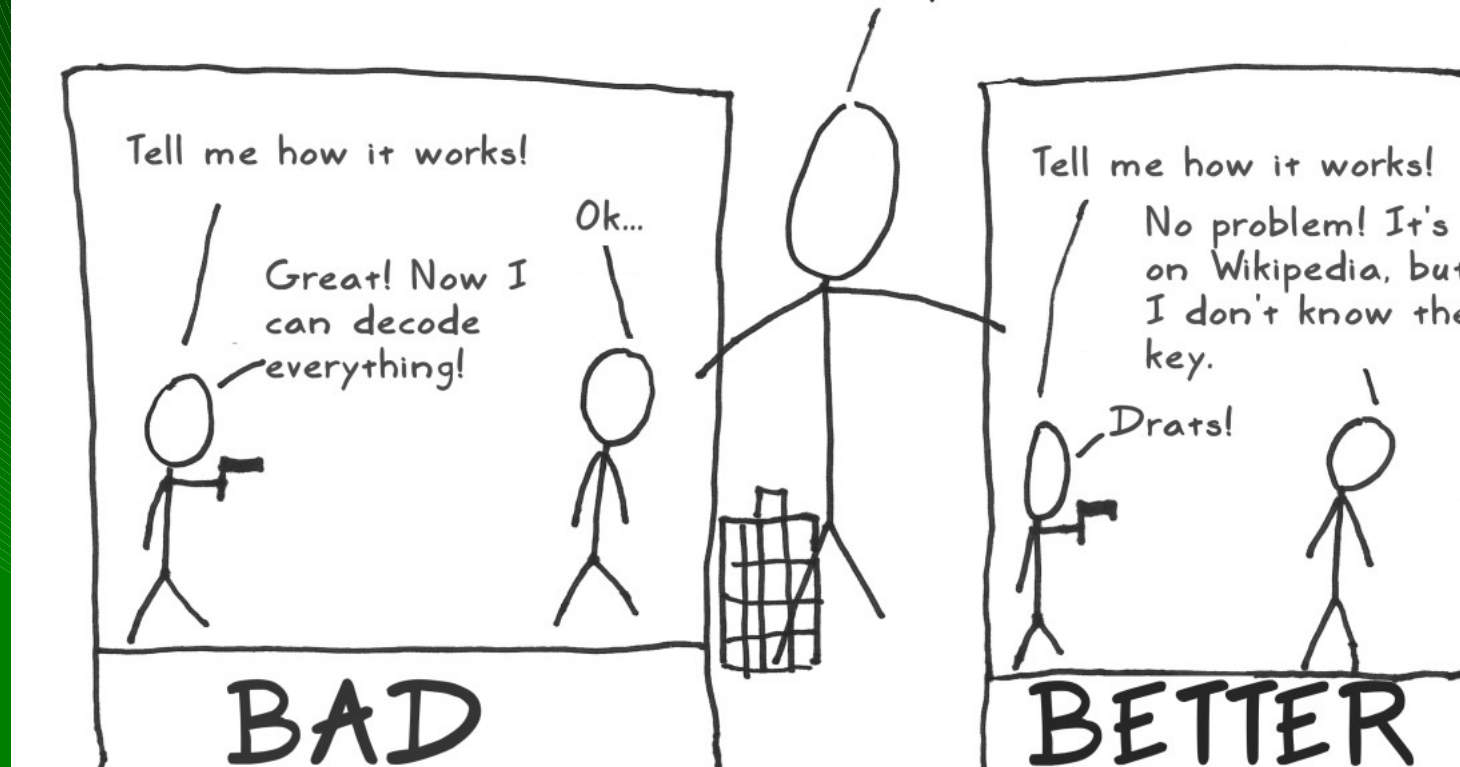
0x30c0 – MDMCFG2 register

0xDF0E: MDMCFG2 - Modem Configuration							
Bit	Field Name	Reset	R/W	Description			
7	DEM_DCFILT_OFF	0	R/W	Disable digital DC blocking filter before demodulator. The recommended IF frequency changes when the DC blocking is disabled. Please use SmartRF® Studio [9] to calculate correct register setting.			
				0	Enable	Better Sensitivity	
				1	Disable	Current optimized. Only for data rates ≤ 100 kBaud	
6:4	MOD_FORMAT[2:0]	000	R/W	The modulation format of the radio signal			
				000	2-FSK		
				001	GFSK		
				010	Reserved		
				011	ASK/OOK		
				100	Reserved		
				101	Reserved		
				110	Reserved		
				111	MSK		
				Note that MSK is only supported for data rates above 26 kBaud and GFSK, ASK, and OOK is only supported for data rate up until 250 kBaud. MSK cannot be used if Manchester encoding/decoding is enabled.			
3	MANCHESTER_EN	0	R/W	Manchester encoding/decoding enable			
				0	Disable		
				1	Enable		
				Note that Manchester encoding/decoding cannot be used at the same time as using the FEC/Interleaver option or when using MSK modulation.			
2:0	SYNC_MODE[2:0]	010	R/W	Sync-word qualifier mode.			
				The values 000 and 100 disables preamble and sync word transmission in TX and preamble and sync word detection in RX.			
				The values 001, 010, 101 and 110 enables 16-bit sync word transmission in TX and 16-bits sync word detection in RX. Only 15 of 16 bits need to match in RX when using setting 001 or 101. The values 011 and 111 enables repeated sync word transmission in TX and 32-bits sync word detection in RX (only 30 of 32 bits need to match).			
				000	No preamble/sync		
				001	15/16 sync word bits detected		
				010	16/16 sync word bits detected		
				011	30/32 sync word bits detected		
				100	No preamble/sync, carrier-sense above threshold		
				101	15/16 + carrier-sense above threshold		
110	16/16 + carrier-sense above threshold						
111	30/32 + carrier-sense above threshold						

sorry, couldn't resist

Big Idea #3: Secrecy Only in the Key

After thousands of years, we learned that it's a bad idea to assume that no one knows how your method works. Someone will eventually find that out.



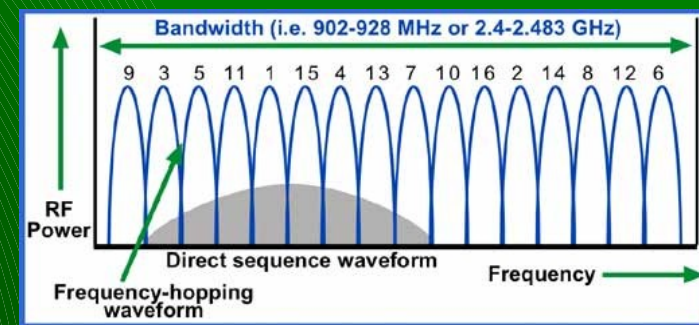
0x3100 – how can we figure it out!?

- open / public documentation
 - insulin pump published frequency
- open source implementation / source code
- “public” but harder to find (google fail!)
 - fcc.gov – search for first part of FCC ID
 - <http://transition.fcc.gov/oet/ea/fccid/> -bookmark it
 - patents – amazing what people will patent!
 - french patent describing the whole MAC/PHY
- reversing hw
 - tapping bus lines – logic analyzer
 - grab config data
 - grab tx/rx data
 - pulling and analyzing firmware

0x3101 – how can we figure it out!? - part 2

- hopping pattern analysis
 - arrays of dongles – space them out and record results
 - hedyattack, or something similar
 - spectrum analyzer
 - USRP2
- trial and error – rf parameters
- MAC layer? - takes true reversing

0x2000 – intro to FHSS



- FHSS is common for devices in the ISM bands
 - provides natural protection against unintentional jamming /interference
 - US Title 47 CFR 15.247 affords special power considerations to FHSS devices
 - >25khz between channels
 - pseudorandom pattern
 - each channel used equally (avg) by each transmitter
 - if 20db of hopping channel < 250khz:
 - must have at least 50 channels
 - average <0.4sec per 20 seconds on one channel
 - if 20dB of hopping channel >250khz:
 - must have at least 25 channels
 - average <0.4sec per 10 seconds on one channel

0x2010 – FHSS, the one and only - NOT!

- different technologies:
 - DSSS – Direct Sequence Spread Spectrum
 - hops happen more often than bytes (ugh)
 - typically requires special PHY layer
 - “FHSS”
 - hops occur after a few symbols are transmitted
- different topologies: (allow for different synch methods)
 - point-to-point (only two endpoints)
 - multiple access systems
- different customers:
 - military has used frequency hopping since Hedy and George submitted the patent in 1941.
 - commercial folks (WiFi, Bluetooth, proprietary stuff like power meters)

0x2020 – FHSS intricacies

- what's so hard about FHSS?
 - must know or be able to come up with the hopping pattern
 - can be anywhere from 50 to a million distinct channel hops before the pattern repeats (or more)
 - must be able to synchronize with an existing cell or partner
 - or become your own master!
 - must know channel spacing
 - must know channel dwell time (time to sit on each channel)
 - likely need to reverse engineer your target
 - DSSS requires that you have special hardware
- military application will be very hard to crack, as it typically will have hops based on a synchronized PRNG to select channels

0x2030 – FHSS, the saving graces

- any adhoc FHSS multi-node network: (power meters / sensor-nets)
 - node sync in a reasonable timeframe
 - limited channels in the repeated pattern
 - each node knows how to talk to a cell
 - let one figure it out, then tap the SPI bus to see what the pattern is...
- two keys to determining hopping pattern:
 - hop pattern generation algorithm
 - typically based on the CELL ID
 - one pattern gets you the whole cell :)
 - some sync information the cell gives away for free
 - gotta tell the n00bs how to sync up, right?
 - for single-pass repeating sequences, it's just the channel

0x2040 – FHSS summary

- FHSS comes in different forms for different uses and different users
- FHSS is naturally tolerant to interference, and allows a device to transmit higher power than nonFHSS comms
- getting the FHSS pattern, timing, and appropriate sync method for proprietary comms can be a reversing challenge
- getting a NIC to do something with the knowledge gained above has – to date – been very difficult

0x3000 – intro to the RFCAT project

- rfcats: RF Chipcon-based Attack Toolset
- background...
- goals...
- plans...
- where we're at so far...

0x3010 – rfcats background

- the power grid
 - power meters and the folks who love them (yo cutaway, q, travis and josh!)
 - no availability of good attack tools for RF
- vendor at Distributech 2008:
 - “Our Frequency Hopping Spread Spectrum is too fast for hackers to attack.”
 - OMFW! really?

0x3020 – rfcats goals

- RE tools - “how does this work?”
 - security analysis tools - “your FHSS and Crypto is weak!”
 - satiate my general love of RF
-
- a little of Nevil Maskelyne
 - “I will not demonstrate to any man who throws doubt upon the system” - Guglielmo Marconi, 1903

0x3030 – this is not HedyAttack

- but leveraged the knowledge from HA...
- this is the base code which HedyAttack started...
- less "researchy" (this project won't find hopping patterns)
- more utilitarian - give us comms parameters and a hopping pattern, and we'll be a NIC, sniffer, and interact with RF gadgets

0x3040 – rfcats interface

- rfcats is many things, but I like to think of it as an interactive python access to the <GHz spectrum!
 - <insert pic>
- rfcats_nic.py
 - FHSS-capable NIC (some assembly may be required for FHSS to arbitrary devices)
 - toolset for discovering what is currently using a given band/channel
- rfcats_nic_server.py
 - access the <GHz band over an IP network or locally and configure on the fly

0x3050 – rfcats_nic.py

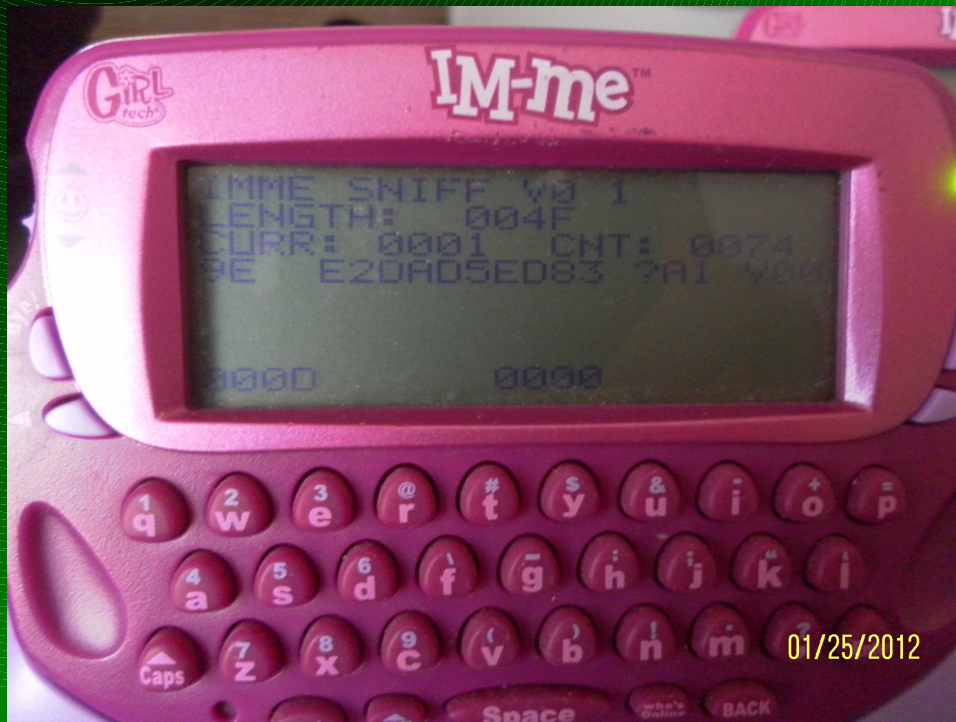
- customizable NIC-access to the ISM bands
- ipython for best enjoyment
- <insert pic>
- lame spoiler: you get a global object called “d” to talk to the dongle
- `d.RFxmit('blah')`
- `data = d.RFrecv()`
- `d.discover(lowball=1)`

0x3060 – rfcats_nic_server.py

- bringing <GHz over the IP network!
- connect on one TCP port to access the wireless network
- connect on a second TCP port to access the wireless configuration
- created to allow non-python clients to play too
 - stdin is not the way you want to interact with embedded wireless protocols
 - <insert pic>

0x3070 – rfsniff (pink version too!)

- focused primarily on capturing data from the wireless network
- IMME used to provide a nice simple interface
- recently added RF config adjustment using keyboard!



0x3100 – one dongle to rule them all

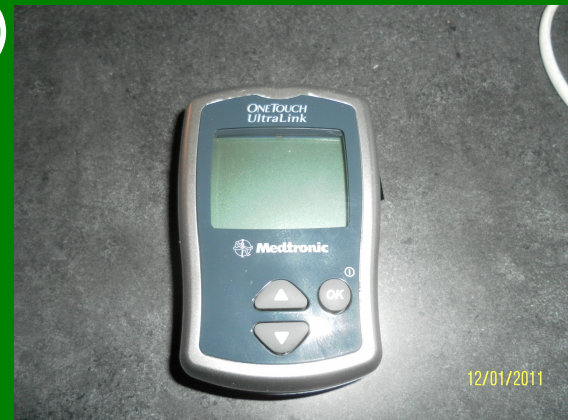
- example RF attack lab setup:
 - dongle “Gina” running hedyattack spec-an code
 - dongle “Anna” running rfcats_nic.py
 - IMME running rfsniff
 - (possibly an IMME's running SpecAn)
 - saleae logic analyzer for hacking of the wired variety

rf attack form

- base freq:
- modulation:
- baud/bandwidth:
- deviation:
- channel hopping?
 - how many channels: channel spacing:
 - pattern and effective sync method?
 - dwell period (ms):
- fixed-/variable-length packets: len/maxlen:
- “address”:
- sync word (if applicable):
- crc16 (y/n): does chip do correct style?
- fec (y/n): type (convolutional/reed-soloman/other):
- manchester encoding (y/n):
- data whitening? and 9bit pattern:
-

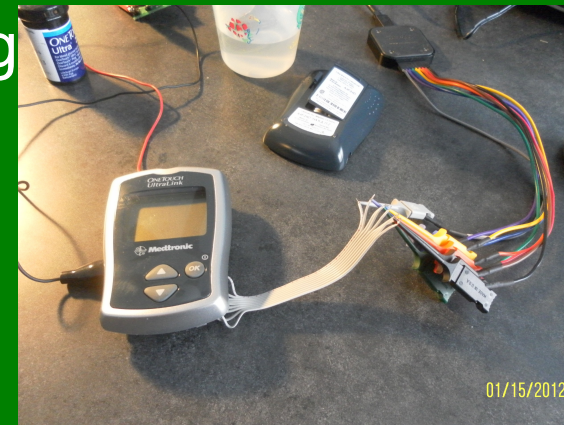
0x4000 – playing with a medical device

- **CAUTION: MUCKING WITH THESE CAN KILL PEOPLE.**
 - THIS FIRMWARE AND CLIENT NOT PROVIDED
- found frequency in the pdf manual from the Internet
 - what random diabetic cares what frequency his pump communicates with!? ok, who cares!
- modulation guessed based on spectrum analysis and trial/error
 - the wave form just **looks** like <blah> modulation!
- other characteristics discovered using a USRP and baudline (and some custom tools, thanks Mike Ossman!)

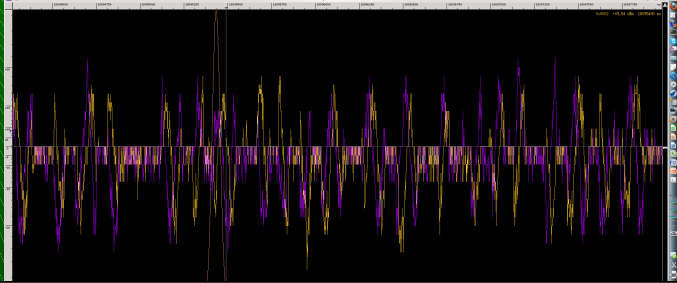


0x4010 – the discovery process

- glucometer was first captured using Spectrum Analyzer (IMME/hedyattack) to validate frequency range from the lay-documentation
- next a logic analyzer (saleae) used to tap debugging lines
- next, the transmission was captured using a USRP (thank you Mike Ossman for sending me your spare!)
- next, the “packet capture” was loaded into Baudline, and analysis performed to identify baudrate and modulation scheme, and get an idea of bits
- next, Mike Ossman did amazing-sauce, running the capture through GnuRadio Companion (the big picture on next slide)
- RF parameters confirmed through RF analysis, and real-life capture.



0x4011 – discovery reloaded

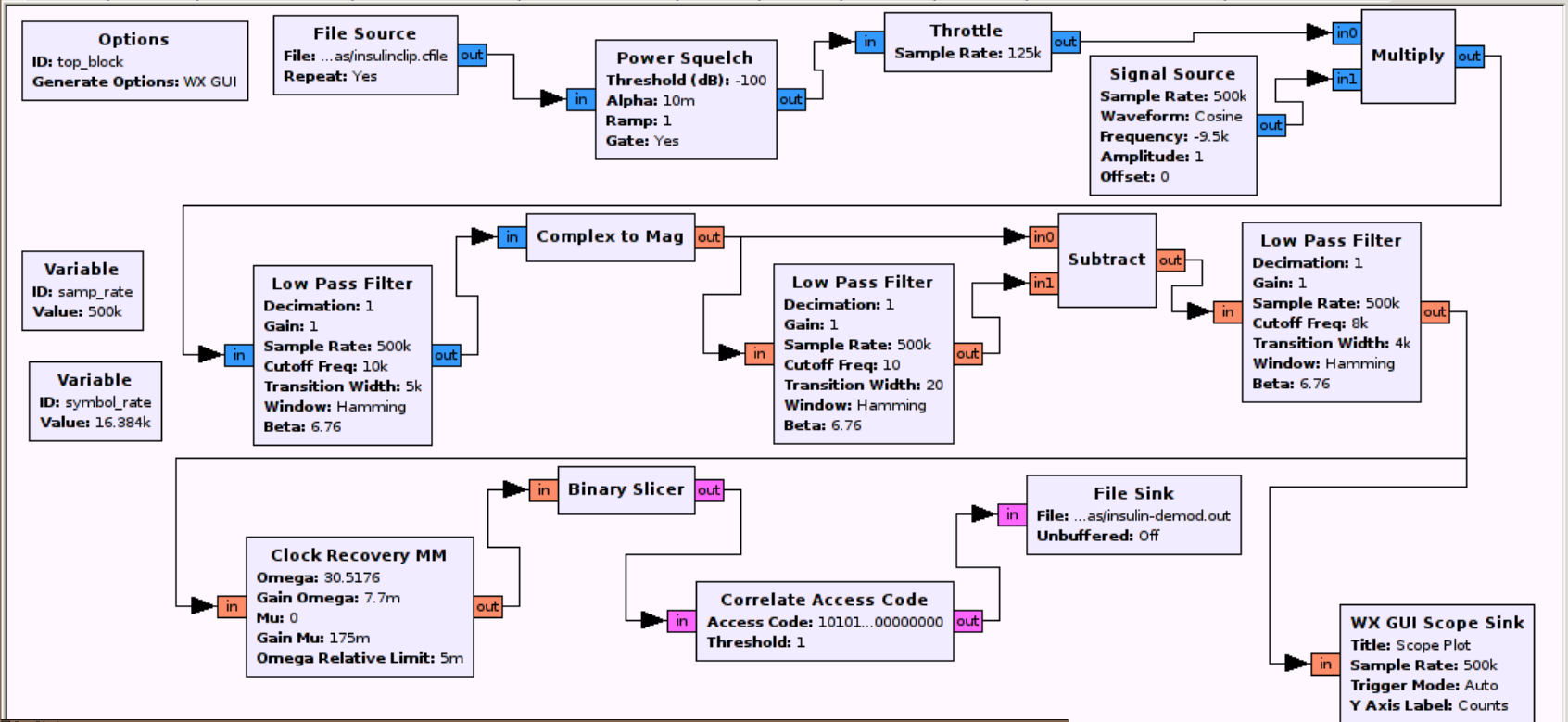


insulin-demod.grc - /home/mossmann/shmoo/2012/atlas - GNU Radio Companion

File Edit View Build Help



ex6 X filter X xyloc-demo X xyloc-demod X xyloc-replay X ex1 X ex8 X ex8b X ex8c X docsis-interferer X atlas-demod X insulin-demod X



- Blocks
 - [Sources]
 - [Sinks]
 - [Operators]
 - [Type Conversions]
 - [Stream Conversions]
 - [Misc Conversions]
 - [Synchronizers]
 - [Level Controls]
 - [Filters]
 - [Modulators]
 - [Error Correction]
 - [Line Coding]
 - [Vocoders]
 - [Probes]
 - [Variables]
 - [Misc]
 - [Digital]
 - Binary Slicer
 - Clock Recovery MM
 - CMA Equalizer
 - Constellation Decoder
 - Correlate Access Code
 - Costas Loop
 - FLL Band-Edge
 - Kurtotic Equalizer
- + Add



0x4100 – playing with a power meter



- **CAUTION:** MUCKING WITH POWER SYSTEMS WITHOUT APPROPRIATE AUTHORIZATION IS ILLEGAL, EVEN IF IT IS ON THE SIDE OF YOUR HOUSE!
- most power meters use their own proprietary “Neighborhood Area Network” (NAN), typically in the 900MHz range and sometimes 2.4GHz or licensed spectrum.
- to get the best reception over distance and gain tolerance to interference, all implement FHSS to take advantage of the Title 47: Part 15 power allowances
- many of the existing meters use the same cc1111 or cc1110 chips, or the cc1101 radio core
- this is the reason I'm here today



0x4110 – as sands through the hourglass

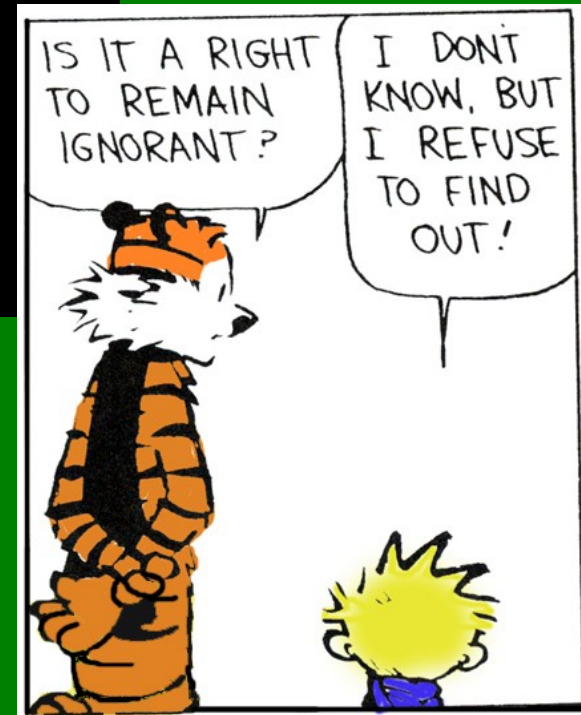
- power meters have long been “unavailable” for most security researchers
- **some vendors understand the benefits of security rigor by outside researchers**
 - others, however, do not.
- the gear used in my presentation was given to me by one such company
 - for various reasons, they have asked to remain anonymous, however, their security team has a well founded approach to finding out “how their baby is ugly” I would like to give them credit for their commitment to the improved security of their products.

atlas, tell us what you really think



IGNORANCE

When did it become a point of view?

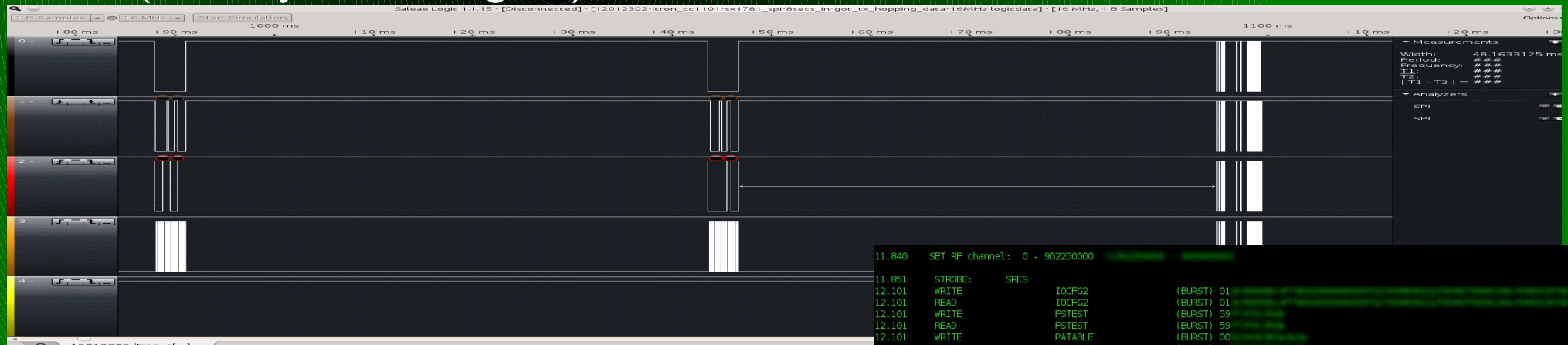


0x4120 – smart meter – the complication

- power meters are not so simple as glucometers
 - proprietary FHSS in a multiple-access topology
 - have to endure the RF abuse of the large metropolis
- complex mac sync/net-registration
- not easy to show with a single meter without a Master node.
- initial analysis was performed via my saleae LA:
- SpecAn code on IMME's and hedyattack dongles
 - good for identifying periods of scanning
- although the dongle can hop along with the meter, we won't be demoing synching with the meter today

0x4130 – the approach

- determine the rf config and hopping pattern through SPI Bus sniffing (and my saleae again)



- decoding:

custom parser for the
target radio--->>>



```
11.840 SET RF channel: 0 - 902250000
11.851 STROBE: SPES
12.101 WRITE IOCFG2 (BURST) 01
12.101 READ IOCFG2 (BURST) 01
12.101 WRITE FSTEST (BURST) 59
12.101 READ FSTEST (BURST) 59
12.101 WRITE PATABLE (BURST) 00
12.103 READ PATABLE (BURST) 00
12.103 STROBE: SIDLE
12.104 STROBE: SCAL
12.104 STROBE: SFRX
12.105 STROBE: SRX
12.107 STROBE: SIDLE
12.108 STROBE: SCAL
12.108 STROBE: SFRX
12.108 STROBE: SRX
12.123 STROBE: SPES
12.373 WRITE IOCFG2 (BURST) 01
12.374 READ IOCFG2 (BURST) 01
12.374 WRITE FSTEST (BURST) 59
12.374 READ FSTEST (BURST) 59
12.375 WRITE PATABLE (BURST) 00
12.375 READ PATABLE (BURST) 00
12.376 STROBE: SIDLE
12.376 STROBE: SCAL
12.377 STROBE: SFRX
12.377 STROBE: SRX
12.394 STROBE: SIDLE
12.395 STROBE: SCAL
12.395 STROBE: SFRX
12.396 STROBE: SRX
12.399 STROBE: SIDLE
12.399 STROBE: SCAL
12.400 STROBE: SFRX
12.400 STROBE: SRX
12.404 STROBE: SIDLE
12.404 STROBE: SCAL
12.405 STROBE: SFRX
12.405 STROBE: SRX
12.409 STROBE: SIDLE
12.409 STROBE: SCAL
12.410 STROBE: SFRX
12.410 STROBE: SRX
12.412 STROBE: SIDLE
12.412 STROBE: SFTX
12.412 STROBE: SPSTXON
12.412 STROBE: STX
12.414 WRITE TXFIFO (BURST) 4:
12.425 STROBE: SIDLE
12.425 STROBE: SCAL
```


0x4150 – the result

“Abuse is no argument”

- Nevil Maskelyne

conclusions

- rfcats discover mode roxors
- rfcats is a **foundation** for your attack tool
 - way more than just a tool in itself
-

References

- <http://rfcat.googlecode.com>
- [http://en.wikipedia.org/wiki/Part_15_\(FCC_rules\)](http://en.wikipedia.org/wiki/Part_15_(FCC_rules))
- http://en.wikipedia.org/wiki/ISM_band
- <http://www.ti.com/lit/ds/swrs033g/swrs033g.pdf> - “the” manual
- http://edge.rit.edu/content/P11207/public/CC1111_USB_HW_User_s_Guide.pdf
- <http://www.ti.com/litv/pdf/swru082b>
- <http://www.ti.com/product/cc1111f32#technicaldocuments>
- <http://www.ti.com/lit/an/swra077/swra077.pdf>
- <http://www.newscientist.com/article/mg21228440.700-dotdashdiss-the-gentleman-hackers-1903-lulz.html>
- <http://saleae.com/>
- <http://zone.ni.com/devzone/cda/epd/p/id/5150> - FSK details (worthwhile!)
- http://www.radagast.org/~dplatt/hamradio/FARS_presentation_on_modulation.pdf
 - very good detailed discussion on deviation/modulation
- http://en.wikipedia.org/wiki/Frequency_modulation

Oxgreetz

- power hardware folk who play nice with security researchers
- cutaway and q (awesome hedyattackers)
- gerard van den bosch
- travis and mossman
- sk0d0 and the four J's
- invisigoth and kenshoto
- Jewel, bug, ringwraith, diva
- Jesus Christ