OPENING CLOSED SYSTEMS WITH GLITCHKIT

34TH CHAOS COMMUNICATION CONGRESS

KATE TEMKIN & DOMINIC SPILL

WHO WE ARE

Kate Temkin

@ktemkin

Major projects:

- FaceDancer
- GreatFET

Dominic Spill

@dominicgs

Major projects:

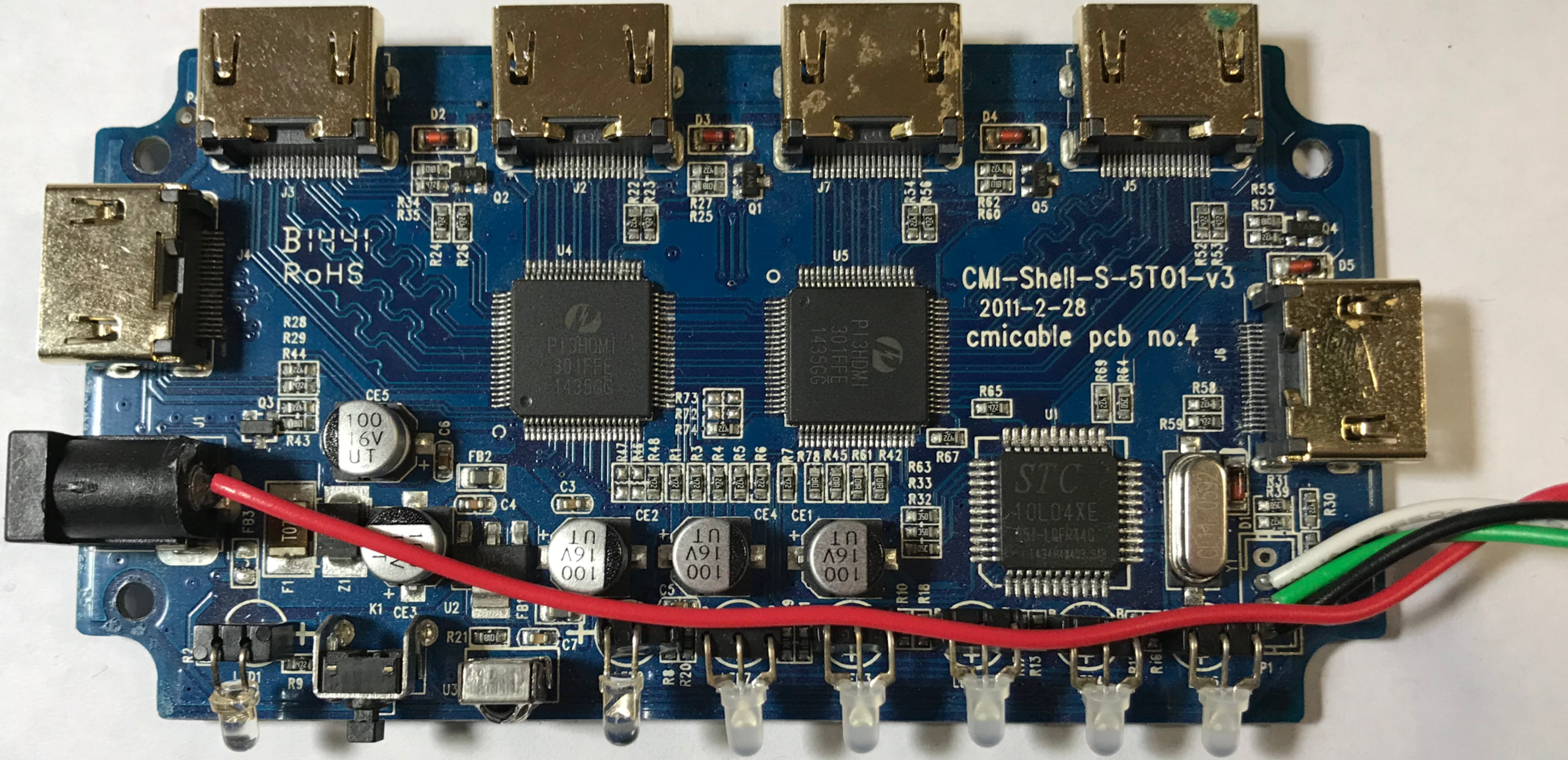
- HackRF
- GreatFET

PEOPLE SMARTER THAN US

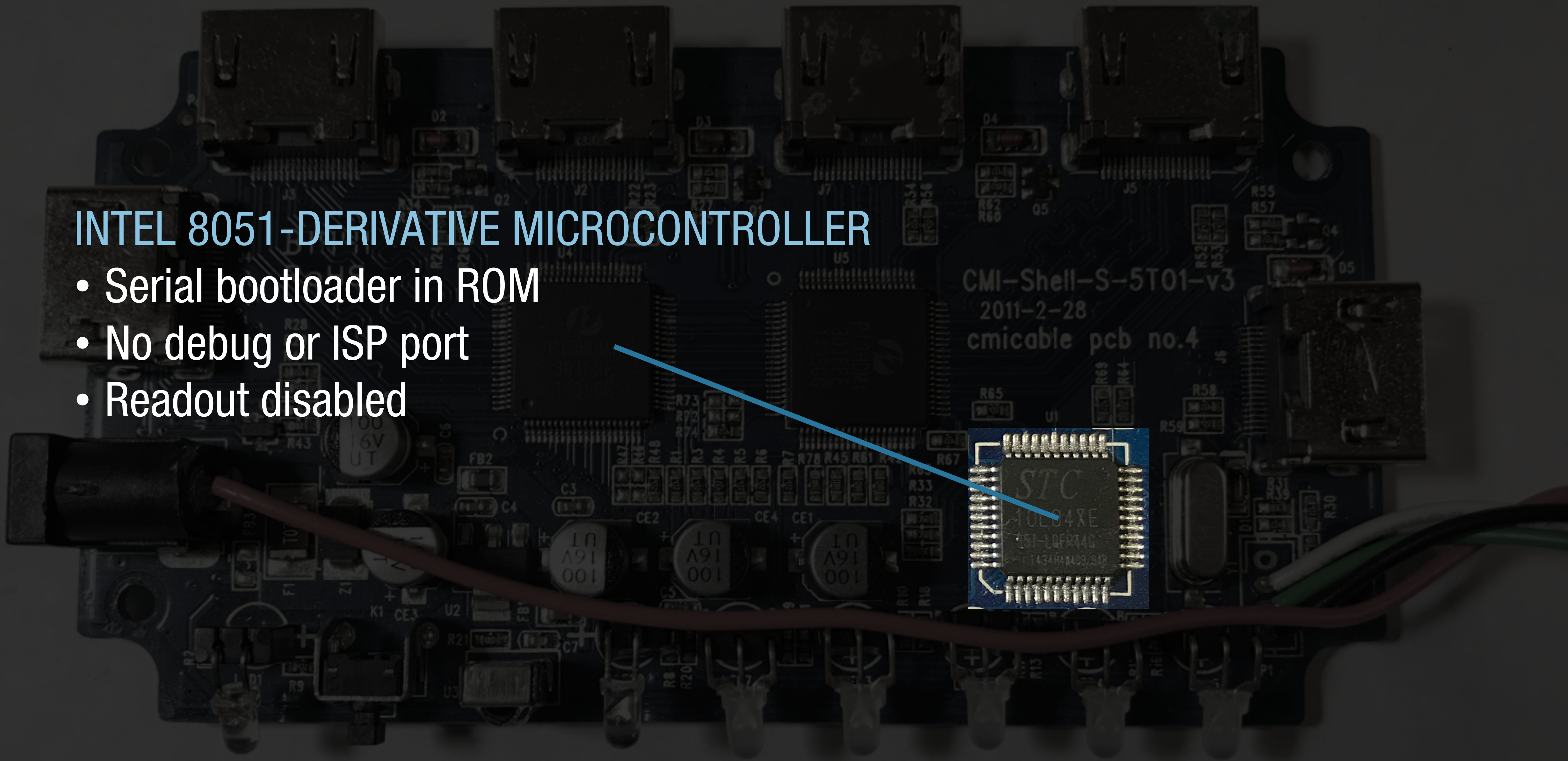
- Micah Elizabeth Scott (@scanlime)
- Colin O’Flynn (@colinoflynn)
- Most of the people in this room!

PEOPLE WHO GIVE US MONEY

- Great Scott Gadgets [thanks, Mike!]



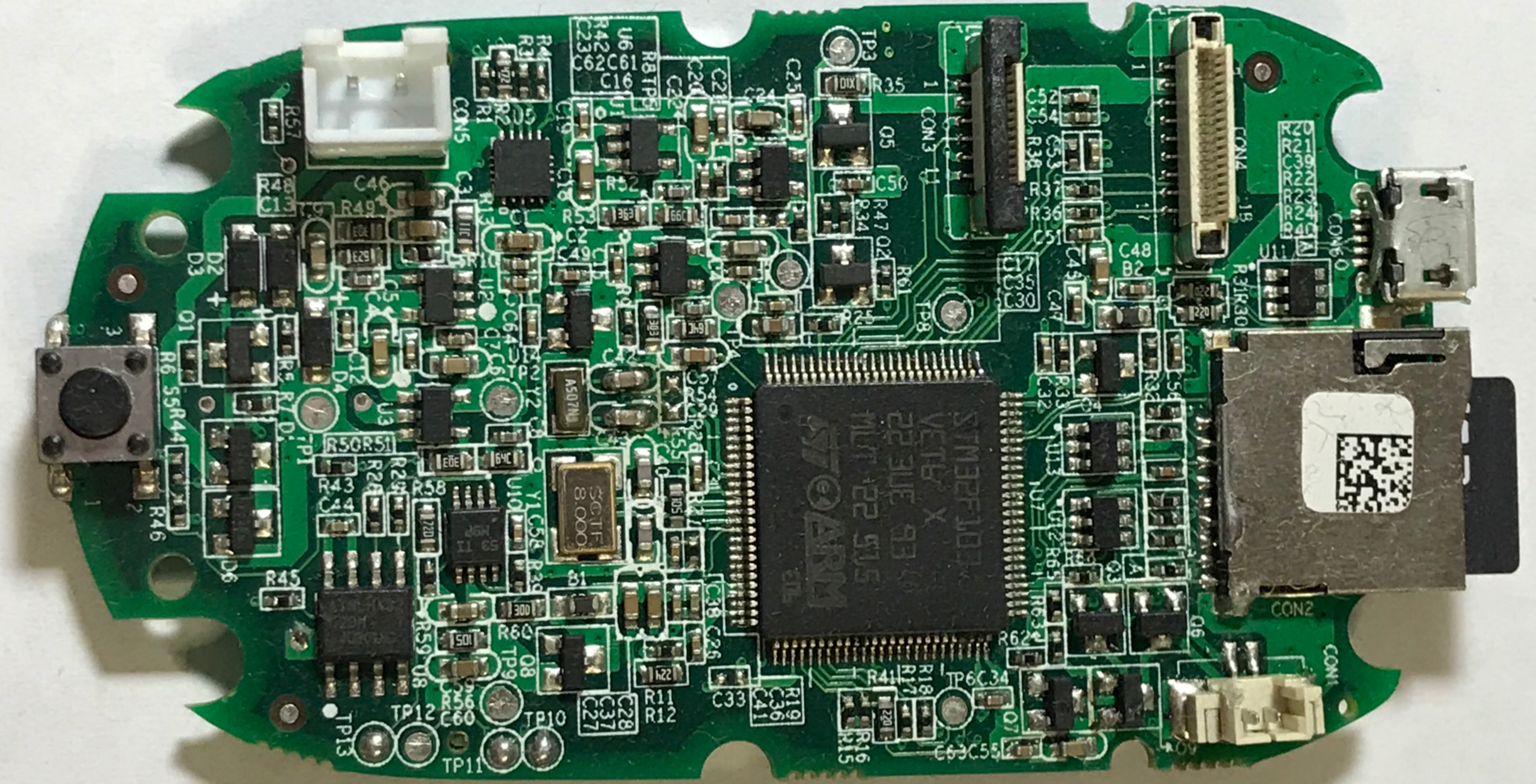
CMI-Shell-S-5T01-v3
2011-2-28
cmicable pcb no.4
BIN41
RoHS
PI3HDMI
301FFE
1435GG
STC
10L04XE
J1
J2
J3
J4
J5
J6
J7
U1
U2
U3
U4
U5
Q1
Q2
Q3
Q4
Q5
D2
D3
D4
D5
CE1
CE2
CE3
CE4
CE5
FB2
FB3
F1
K1
P1
100
16V
UT

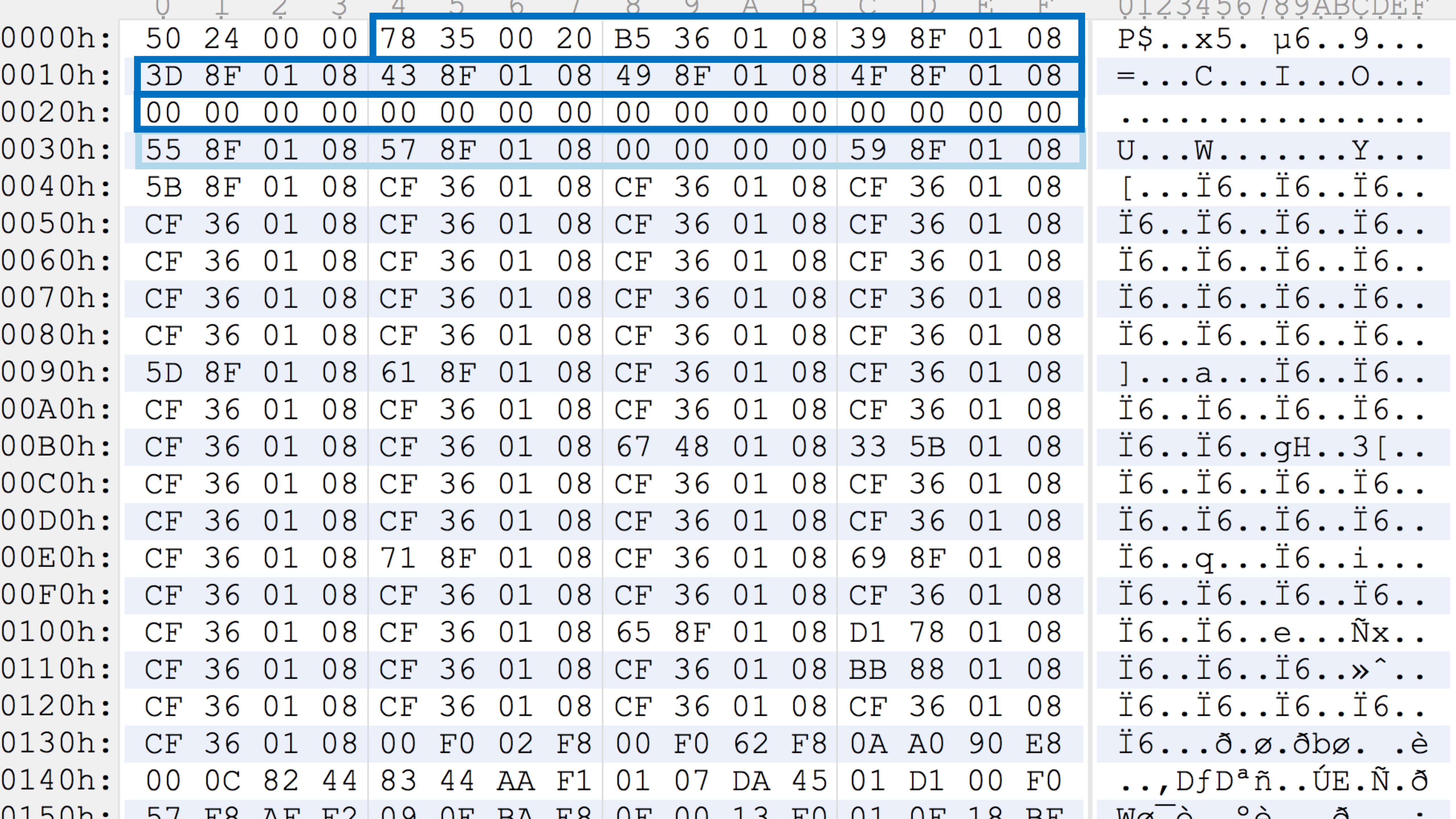


INTEL 8051-DERIVATIVE MICROCONTROLLER

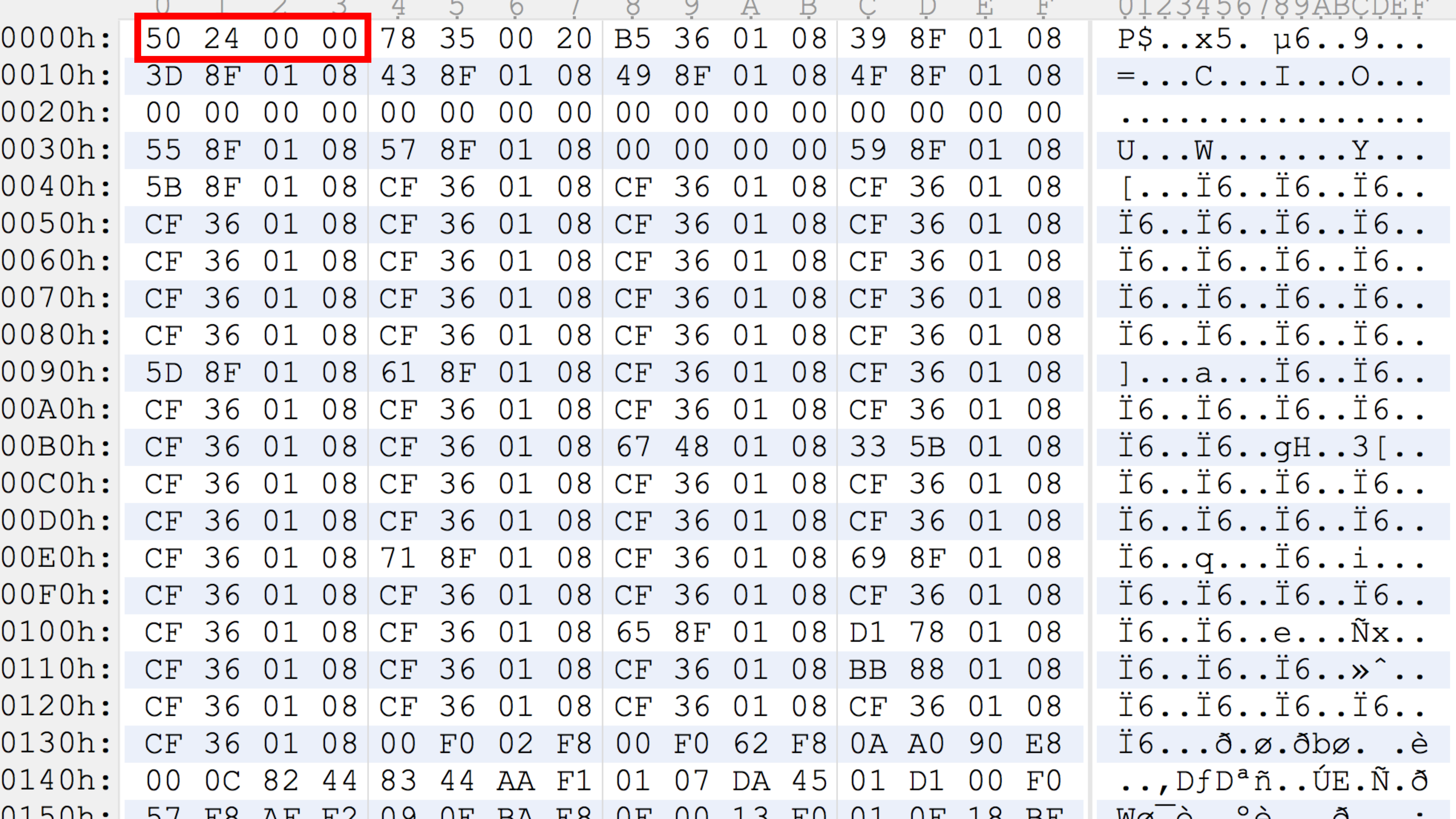
- Serial bootloader in ROM
- No debug or ISP port
- Readout disabled

FLIR TG-165 THERMAL CAMERA

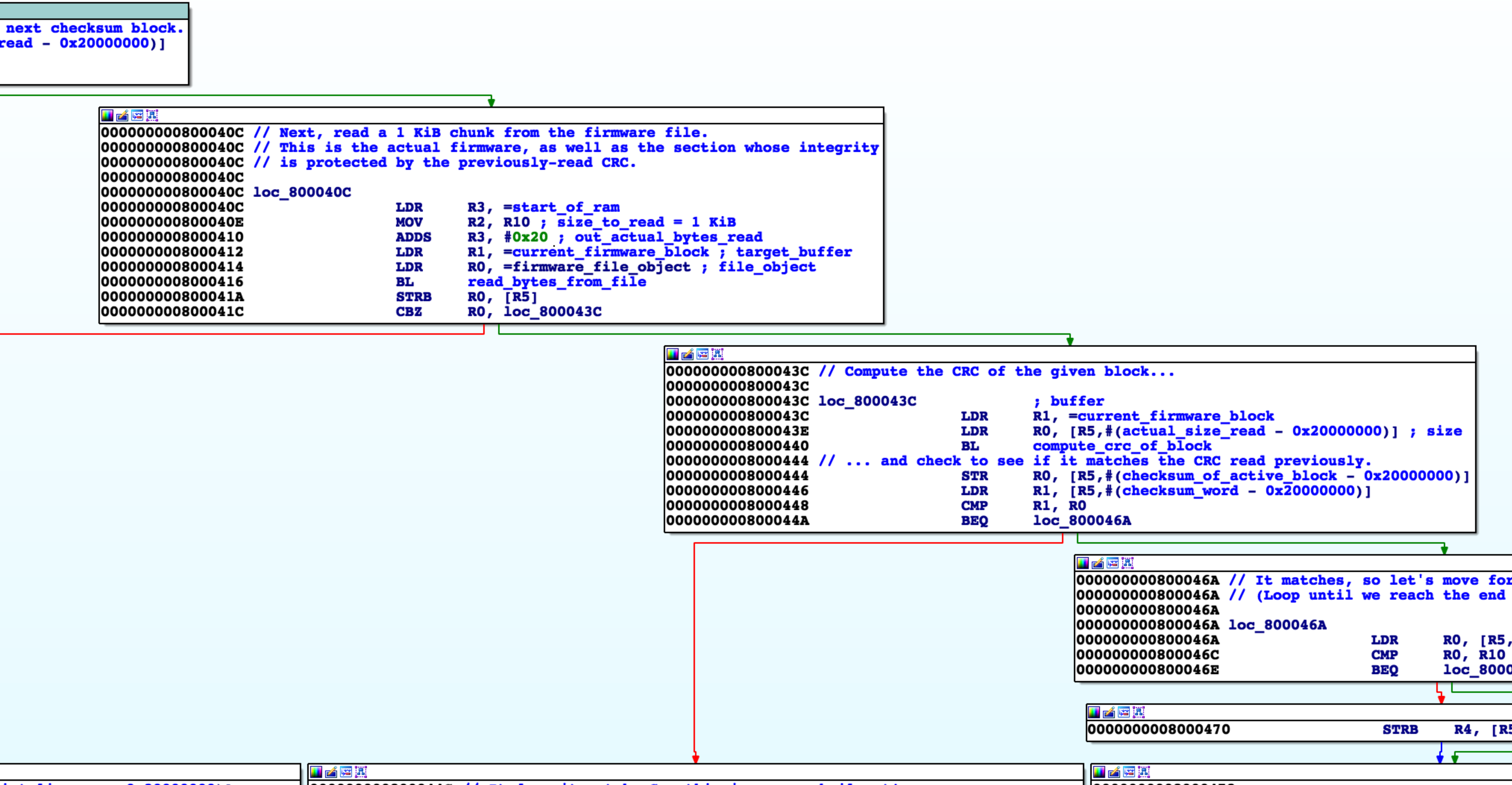




```
        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F    0123456789ABCDEF
0000h: 50 24 00 00 78 35 00 20 B5 36 01 08 39 8F 01 08   P$..x5. µ6..9...
0010h: 3D 8F 01 08 43 8F 01 08 49 8F 01 08 4F 8F 01 08   =...C...I...O...
0020h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0030h: 55 8F 01 08 57 8F 01 08 00 00 00 00 59 8F 01 08   U...W.......Y...
0040h: 5B 8F 01 08 CF 36 01 08 CF 36 01 08 CF 36 01 08   [...Ï6..Ï6..Ï6..
0050h: CF 36 01 08 CF 36 01 08 CF 36 01 08 CF 36 01 08   Ï6..Ï6..Ï6..Ï6..
0060h: CF 36 01 08 CF 36 01 08 CF 36 01 08 CF 36 01 08   Ï6..Ï6..Ï6..Ï6..
0070h: CF 36 01 08 CF 36 01 08 CF 36 01 08 CF 36 01 08   Ï6..Ï6..Ï6..Ï6..
0080h: CF 36 01 08 CF 36 01 08 CF 36 01 08 CF 36 01 08   Ï6..Ï6..Ï6..Ï6..
0090h: 5D 8F 01 08 61 8F 01 08 CF 36 01 08 CF 36 01 08   ]...a...Ï6..Ï6..
00A0h: CF 36 01 08 CF 36 01 08 CF 36 01 08 CF 36 01 08   Ï6..Ï6..Ï6..Ï6..
00B0h: CF 36 01 08 CF 36 01 08 67 48 01 08 33 5B 01 08   Ï6..Ï6..gH..3[..
00C0h: CF 36 01 08 CF 36 01 08 CF 36 01 08 CF 36 01 08   Ï6..Ï6..Ï6..Ï6..
00D0h: CF 36 01 08 CF 36 01 08 CF 36 01 08 CF 36 01 08   Ï6..Ï6..Ï6..Ï6..
00E0h: CF 36 01 08 71 8F 01 08 CF 36 01 08 69 8F 01 08   Ï6..q...Ï6..i...
00F0h: CF 36 01 08 CF 36 01 08 CF 36 01 08 CF 36 01 08   Ï6..Ï6..Ï6..Ï6..
0100h: CF 36 01 08 CF 36 01 08 65 8F 01 08 D1 78 01 08   Ï6..Ï6..e...Ñx..
0110h: CF 36 01 08 CF 36 01 08 CF 36 01 08 BB 88 01 08   Ï6..Ï6..Ï6..»ˆ..
0120h: CF 36 01 08 CF 36 01 08 CF 36 01 08 CF 36 01 08   Ï6..Ï6..Ï6..Ï6..
0130h: CF 36 01 08 00 F0 02 F8 00 F0 62 F8 0A A0 90 E8   Ï6...ð.ø.ðbø. .è
0140h: 00 0C 82 44 83 44 AA F1 01 07 DA 45 01 D1 00 F0   ..‚DƒDªñ..ÚE.Ñ.ð
0150h: 57 F8 AF F2 09 0F BA F8 0F 00 13 F0 01 0F 18 BF
```



```
        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F    0123456789ABCDEF
0000h: 50 24 00 00 78 35 00 20 B5 36 01 08 39 8F 01 08    P$..x5. µ6..9...
0010h: 3D 8F 01 08 43 8F 01 08 49 8F 01 08 4F 8F 01 08    =...C...I...O...
0020h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
0030h: 55 8F 01 08 57 8F 01 08 00 00 00 00 59 8F 01 08    U...W.......Y...
0040h: 5B 8F 01 08 CF 36 01 08 CF 36 01 08 CF 36 01 08    [...Ï6..Ï6..Ï6..
0050h: CF 36 01 08 CF 36 01 08 CF 36 01 08 CF 36 01 08    Ï6..Ï6..Ï6..Ï6..
0060h: CF 36 01 08 CF 36 01 08 CF 36 01 08 CF 36 01 08    Ï6..Ï6..Ï6..Ï6..
0070h: CF 36 01 08 CF 36 01 08 CF 36 01 08 CF 36 01 08    Ï6..Ï6..Ï6..Ï6..
0080h: CF 36 01 08 CF 36 01 08 CF 36 01 08 CF 36 01 08    Ï6..Ï6..Ï6..Ï6..
0090h: 5D 8F 01 08 61 8F 01 08 CF 36 01 08 CF 36 01 08    ]...a...Ï6..Ï6..
00A0h: CF 36 01 08 CF 36 01 08 CF 36 01 08 CF 36 01 08    Ï6..Ï6..Ï6..Ï6..
00B0h: CF 36 01 08 CF 36 01 08 67 48 01 08 33 5B 01 08    Ï6..Ï6..gH..3[..
00C0h: CF 36 01 08 CF 36 01 08 CF 36 01 08 CF 36 01 08    Ï6..Ï6..Ï6..Ï6..
00D0h: CF 36 01 08 CF 36 01 08 CF 36 01 08 CF 36 01 08    Ï6..Ï6..Ï6..Ï6..
00E0h: CF 36 01 08 71 8F 01 08 CF 36 01 08 69 8F 01 08    Ï6..q...Ï6..i...
00F0h: CF 36 01 08 CF 36 01 08 CF 36 01 08 CF 36 01 08    Ï6..Ï6..Ï6..Ï6..
0100h: CF 36 01 08 CF 36 01 08 65 8F 01 08 D1 78 01 08    Ï6..Ï6..e...Ñx..
0110h: CF 36 01 08 CF 36 01 08 CF 36 01 08 BB 88 01 08    Ï6..Ï6..Ï6..»ˆ..
0120h: CF 36 01 08 CF 36 01 08 CF 36 01 08 CF 36 01 08    Ï6..Ï6..Ï6..Ï6..
0130h: CF 36 01 08 00 F0 02 F8 00 F0 62 F8 0A A0 90 E8    Ï6...ð.ø.ðbø. .è
0140h: 00 0C 82 44 83 44 AA F1 01 07 DA 45 01 D1 00 F0    ..‚DƒDªñ..ÚE.Ñ.ð
0150h: 57 F8 AF F2 09 0F BA F8 0F 00 13 F0 01 0F 18 BF
```



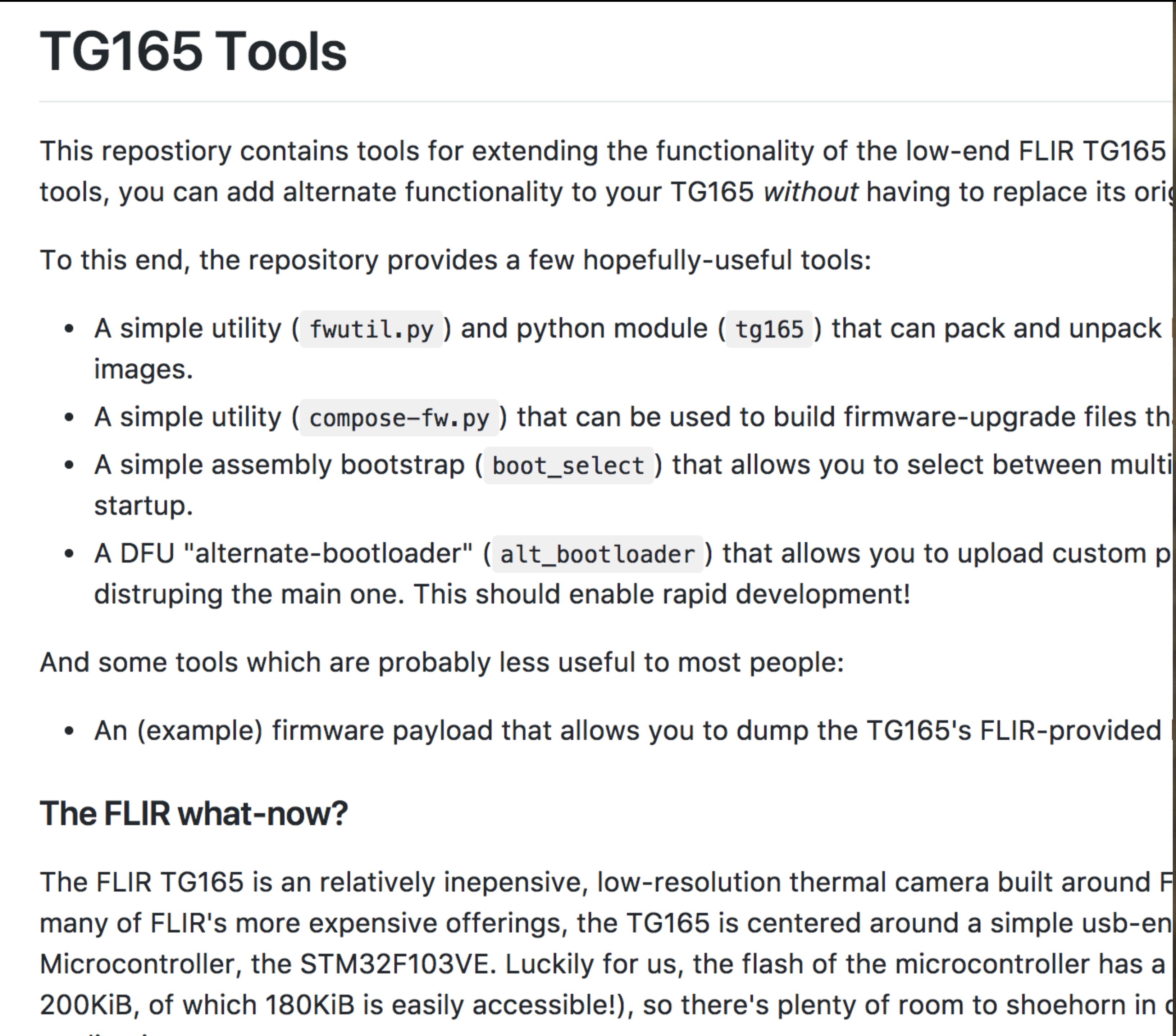
```
 next checksum block.
read - 0x20000000)]
```

```
000000000800040C // Next, read a 1 KiB chunk from the firmware file.
000000000800040C // This is the actual firmware, as well as the section whose integrity
000000000800040C // is protected by the previously-read CRC.
000000000800040C
000000000800040C loc_800040C
000000000800040C                 LDR     R3, =start_of_ram
000000000800040E                 MOV     R2, R10 ; size_to_read = 1 KiB
0000000008000410                 ADDS    R3, #0x20 ; out_actual_bytes_read
0000000008000412                 LDR     R1, =current_firmware_block ; target_buffer
0000000008000414                 LDR     R0, =firmware_file_object ; file_object
0000000008000416                 BL      read_bytes_from_file
000000000800041A                 STRB    R0, [R5]
000000000800041C                 CBZ     R0, loc_800043C
```

```
000000000800043C // Compute the CRC of the given block...
000000000800043C
000000000800043C loc_800043C             ; buffer
000000000800043C                 LDR     R1, =current_firmware_block
000000000800043E                 LDR     R0, [R5,#(actual_size_read - 0x20000000)] ; size
0000000008000440                 BL      compute_crc_of_block
0000000008000444 // ... and check to see if it matches the CRC read previously.
0000000008000444                 STR     R0, [R5,#(checksum_of_active_block - 0x20000000)]
0000000008000446                 LDR     R1, [R5,#(checksum_word - 0x20000000)]
0000000008000448                 CMP     R1, R0
000000000800044A                 BEQ     loc_800046A
```

```
000000000800046A // It matches, so let's move for
000000000800046A // (Loop until we reach the end
000000000800046A
000000000800046A loc_800046A
000000000800046A                 LDR     R0, [R5,
000000000800046C                 CMP     R0, R10
000000000800046E                 BEQ     loc_8000
```

```
0000000008000470                 STRB    R4, [R5
```



TG165 Tools

This repostiory contains tools for extending the functionality of the low-end FLIR TG165
tools, you can add alternate functionality to your TG165 *without* having to replace its orig

To this end, the repository provides a few hopefully-useful tools:

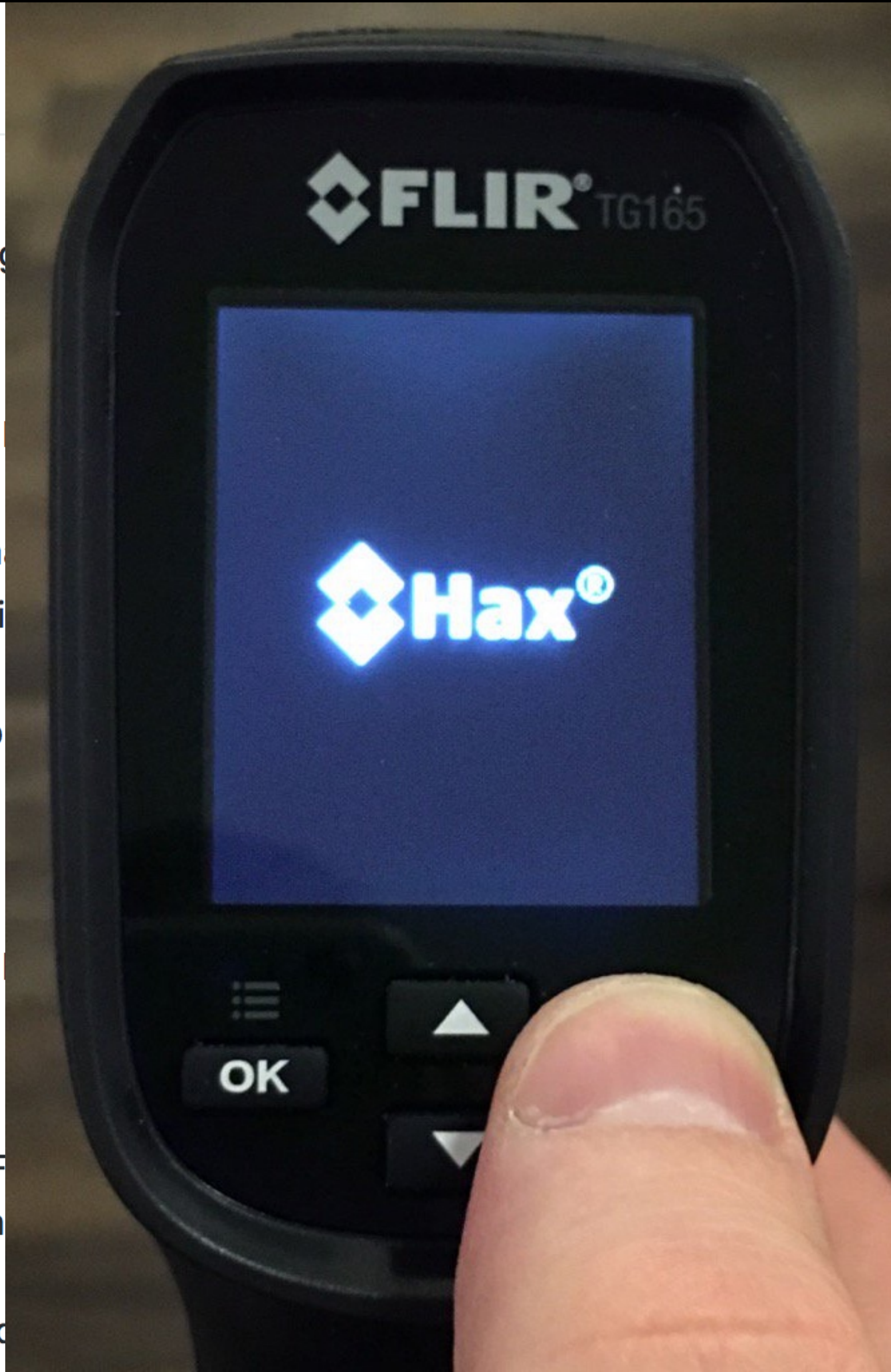
- A simple utility (`fwutil.py`) and python module (`tg165`) that can pack and unpack
  images.
- A simple utility (`compose-fw.py`) that can be used to build firmware-upgrade files th
- A simple assembly bootstrap (`boot_select`) that allows you to select between multi
  startup.
- A DFU "alternate-bootloader" (`alt_bootloader`) that allows you to upload custom p
  distruping the main one. This should enable rapid development!

And some tools which are probably less useful to most people:

- An (example) firmware payload that allows you to dump the TG165's FLIR-provided

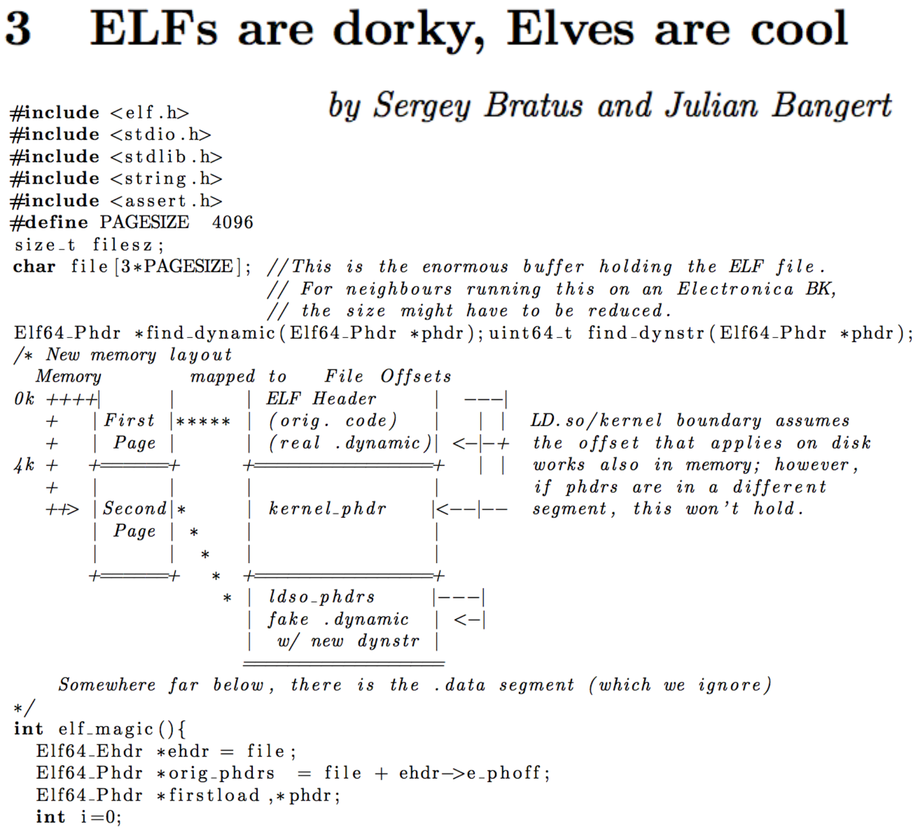
The FLIR what-now?

The FLIR TG165 is an relatively inepensive, low-resolution thermal camera built around F
many of FLIR's more expensive offerings, the TG165 is centered around a simple usb-en
Microcontroller, the STM32F103VE. Luckily for us, the flash of the microcontroller has a
200KiB, of which 180KiB is easily accessible!), so there's plenty of room to shoehorn in c



SECURITY BY NOT MAKING ASSUMPTIONS

```
strcpy(my_stack_memory, user_input);
```



3 ELFs are dorky, Elves are cool

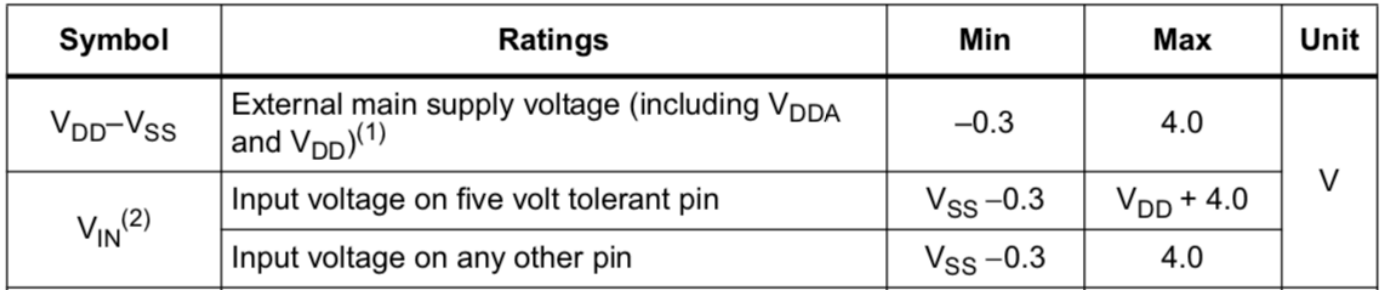
by Sergey Bratus and Julian Bangert

```
#include <elf.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>
#define PAGESIZE  4096
size_t filesz;
char file[3*PAGESIZE]; //This is the enormous buffer holding the ELF file.
                       // For neighbours running this on an Electronica BK,
                       // the size might have to be reduced.
Elf64_Phdr *find_dynamic(Elf64_Phdr *phdr);uint64_t find_dynstr(Elf64_Phdr *phdr);
/* New memory layout
  Memory          mapped to    File Offsets
0k ++++|       |        | ELF Header         |  ---|
   +   | First |*****   | (orig. code)       |   | |  LD.so/kernel boundary assumes
   +   | Page  |        | (real .dynamic)    | <-|-+  the offset that applies on disk
4k +   +=======+        +====================+   | |  works also in memory; however,
   +   |       |        |                    |        if phdrs are in a different
   ++> | Second|*       |  kernel_phdr       |<--|--  segment, this won't hold.
       |  Page | *      |                    |
       |       |  *     |                    |
       +=======+   *    +====================+
                    *   | ldso_phdrs         |---|
                        | fake .dynamic      | <-|
                        |  w/ new dynstr     |
                        ======================

   Somewhere far below, there is the .data segment (which we ignore)
*/
int elf_magic(){
  Elf64_Ehdr *ehdr = file;
  Elf64_Phdr *orig_phdrs  = file + ehdr->e_phoff;
  Elf64_Phdr *firstload,*phdr;
  int i=0;
```

SECURITY BY NOT MAKING ASSUMPTIONS ...?!

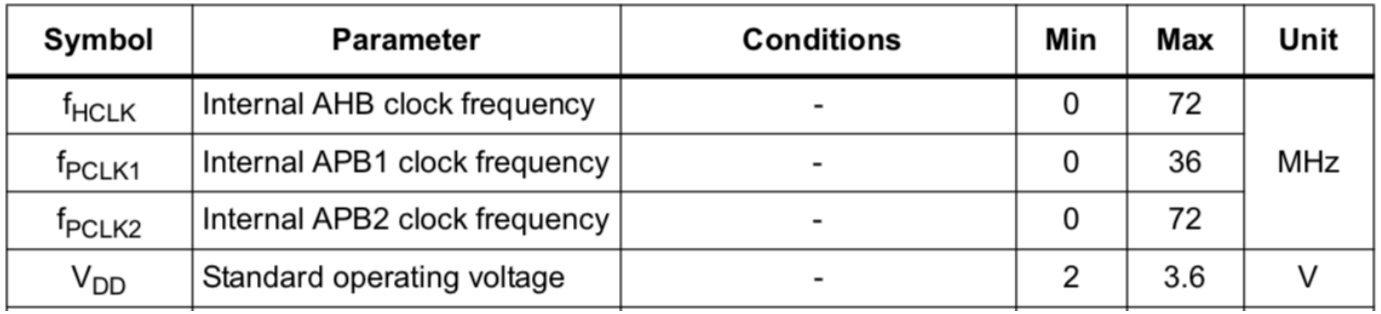
```
strcpy(my_stack_memory, user_input);
```

Table 7. Voltage characteristics

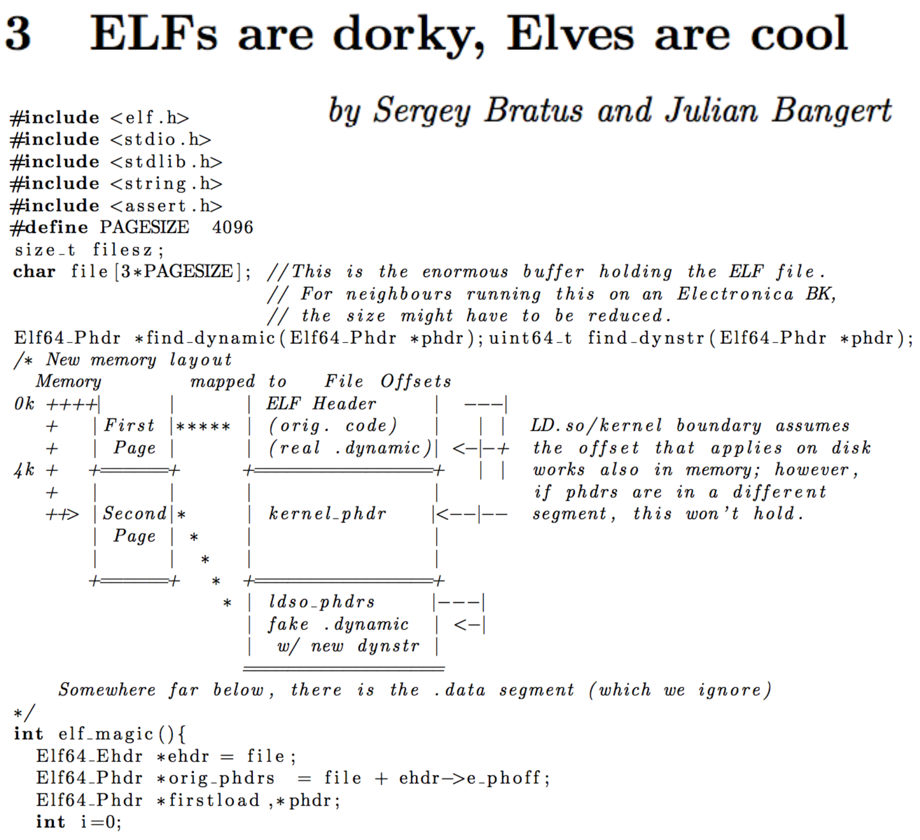


Symbol	Ratings	Min	Max	Unit
$V_{DD}–V_{SS}$	External main supply voltage (including V_{DDA} and V_{DD})$^{(1)}$	–0.3	4.0	V
$V_{IN}^{(2)}$	Input voltage on five volt tolerant pin	V_{SS} –0.3	V_{DD} + 4.0	
	Input voltage on any other pin	V_{SS} –0.3	4.0	

Table 10. General operating conditions



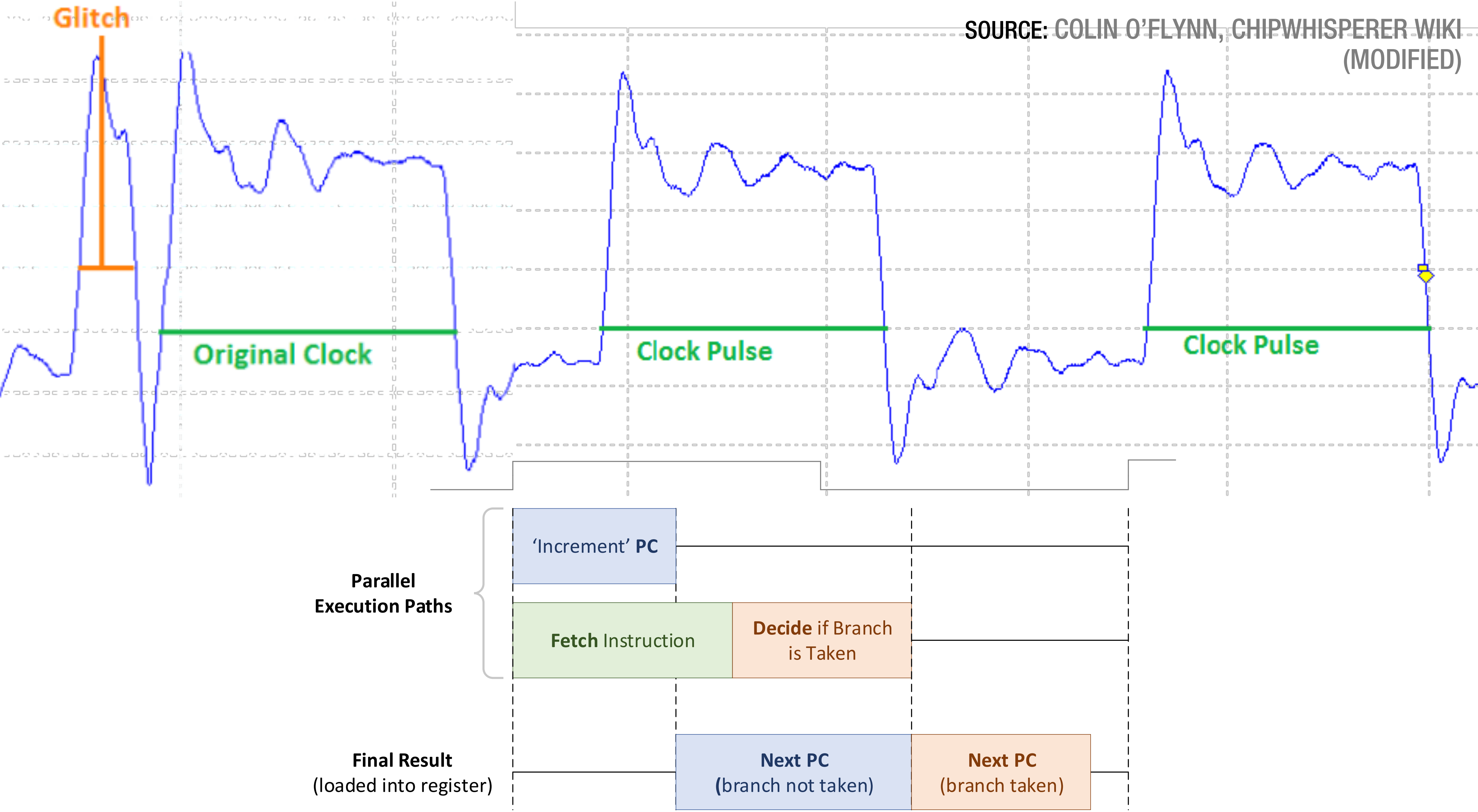
Symbol	Parameter	Conditions	Min	Max	Unit
f_{HCLK}	Internal AHB clock frequency	-	0	72	MHz
f_{PCLK1}	Internal APB1 clock frequency	-	0	36	
f_{PCLK2}	Internal APB2 clock frequency	-	0	72	
V_{DD}	Standard operating voltage	-	2	3.6	V



3 ELFs are dorky, Elves are cool

by Sergey Bratus and Julian Bangert

```
#include <elf.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>
#define PAGESIZE  4096
size_t filesz;
char file[3*PAGESIZE]; //This is the enormous buffer holding the ELF file.
                       // For neighbours running this on an Electronica BK,
                       // the size might have to be reduced.
Elf64_Phdr *find_dynamic(Elf64_Phdr *phdr);uint64_t find_dynstr(Elf64_Phdr *phdr);
/* New memory layout
  Memory          mapped to    File Offsets
0k ++++|        |        | ELF Header         |  ---|
   +   | First  |*****   | (orig. code)       |   | |  LD.so/kernel boundary assumes
   +   |  Page  |        | (real .dynamic)|   <-|-+    the offset that applies on disk
4k +   +========+        +====================+   | |  works also in memory; however,
   +   |        |        |                    |        if phdrs are in a different
   ++> | Second |*       | kernel_phdr        |<--|--  segment, this won't hold.
       |  Page  | *      |                    |
       |        |  *     |                    |
       +========+   *    +====================+
                     *   | ldso_phdrs         |---|
                         | fake .dynamic      | <-|
                         |  w/ new dynstr     |
                         ======================
    Somewhere far below, there is the .data segment (which we ignore)
*/
int elf_magic(){
  Elf64_Ehdr *ehdr = file;
  Elf64_Phdr *orig_phdrs  = file + ehdr->e_phoff;
  Elf64_Phdr *firstload ,*phdr;
  int i=0;
```



SOURCE: COLIN O'FLYNN, CHIPWHISPERER WIKI (MODIFIED)

Parallel Execution Paths

'Increment' **PC**

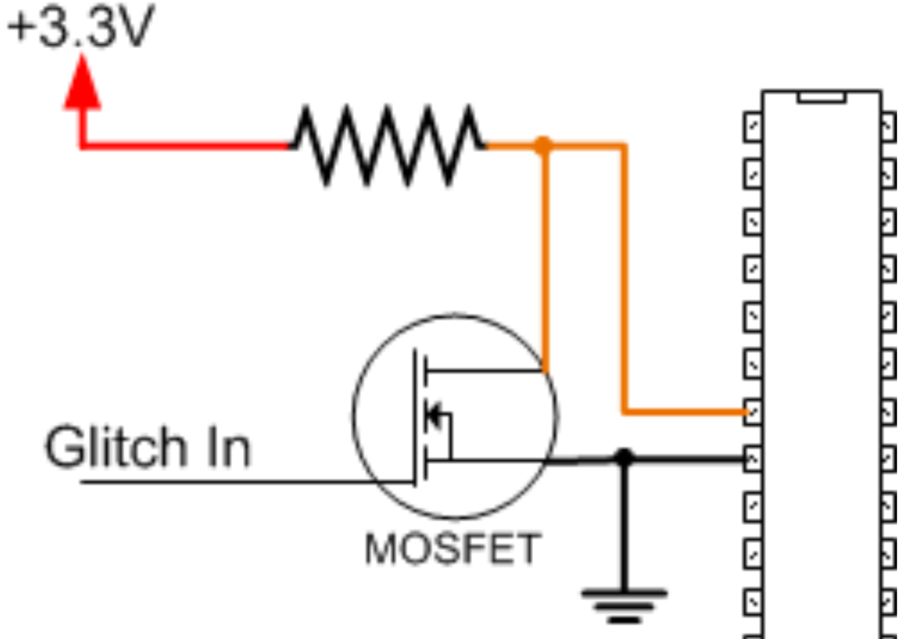
Fetch Instruction

Decide if Branch is Taken

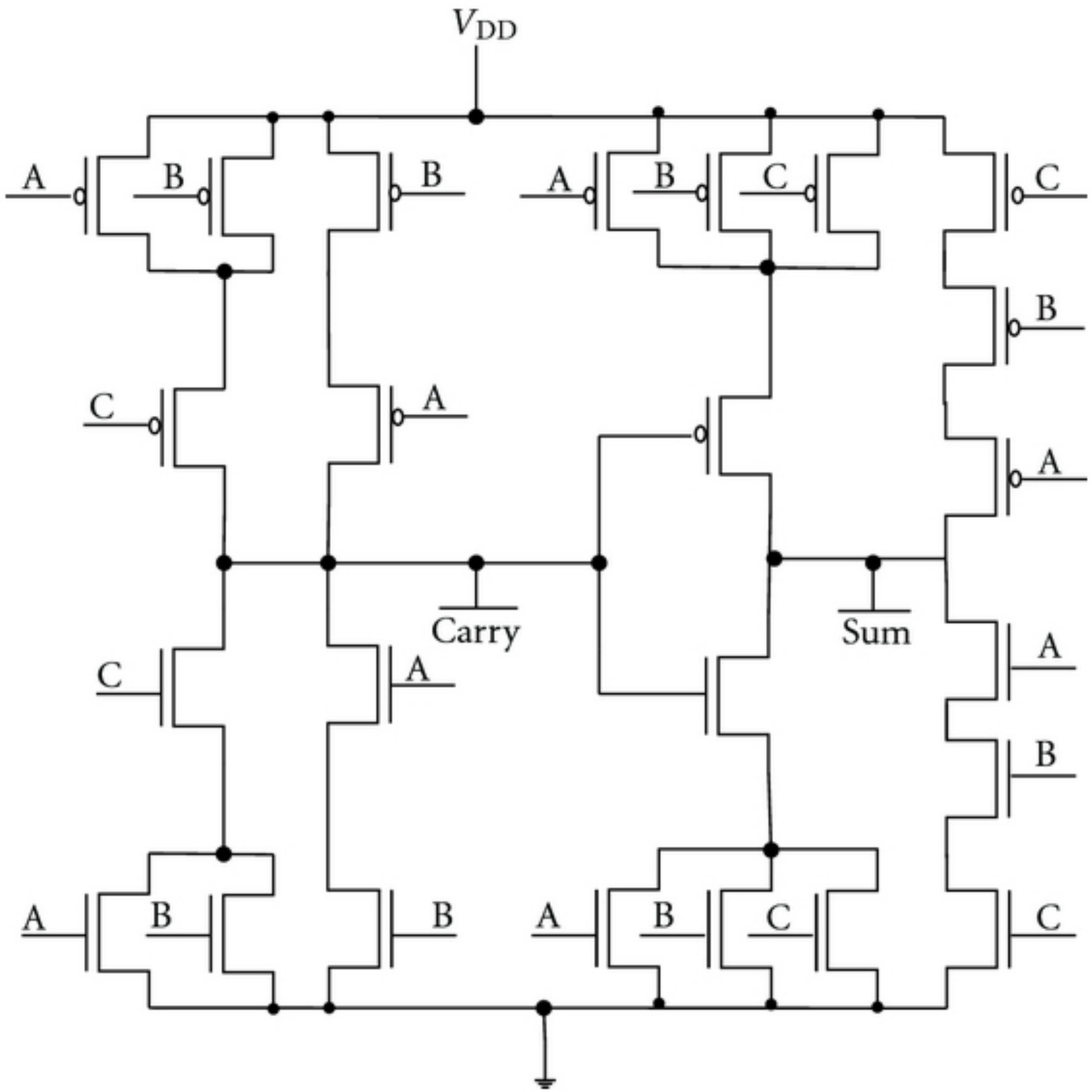
Final Result
(loaded into register)

Next PC
(branch not taken)

Next PC
(branch taken)



SOURCE: COLIN O'FLYNN, CHIPWHISPERER WIKI



SOURCE: NAVI ET AL, LOW-POWER AND HIGH-PERFORMANCE 1-BIT CMOS FULL ADDER CELL

PSEUDOCODE PSEUDO-EXAMPLE

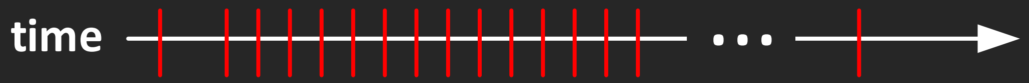
```
raw    = (char *)items;
length = N * sizeof(items[0]);

while (--length) {
    send_byte(raw++);
}
```

```
  ; [snip]
  ; compute length
  MUL  R1, R11, R12

loop:
  DEC  R1, R1 ; --length
  JZ   finish
  CALL send_byte
  INC  R2, R2 ; raw++
  JMP  loop

finish:
  NOP
```

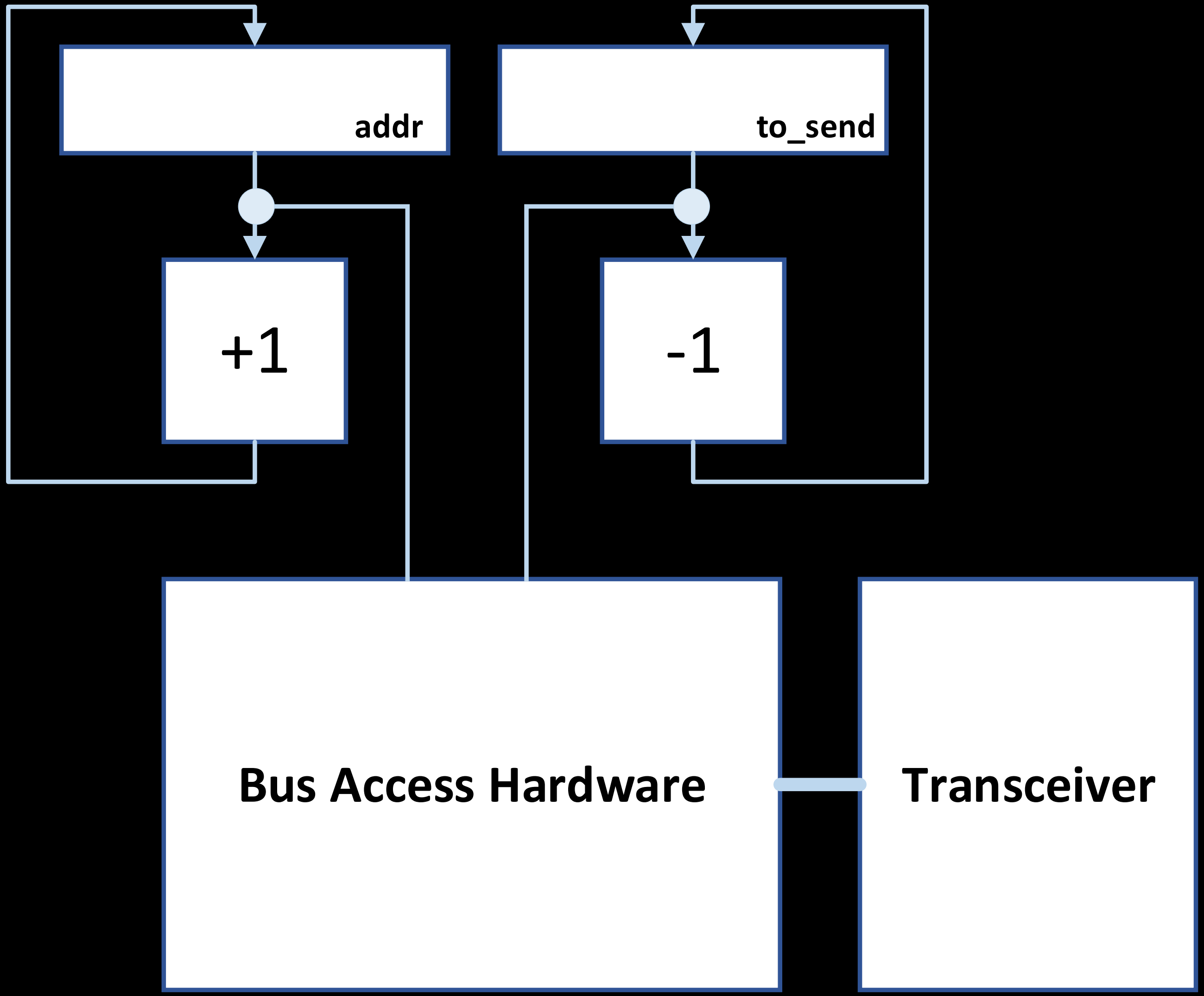
PSEUDOCODE PSEUDO-EXAMPLE

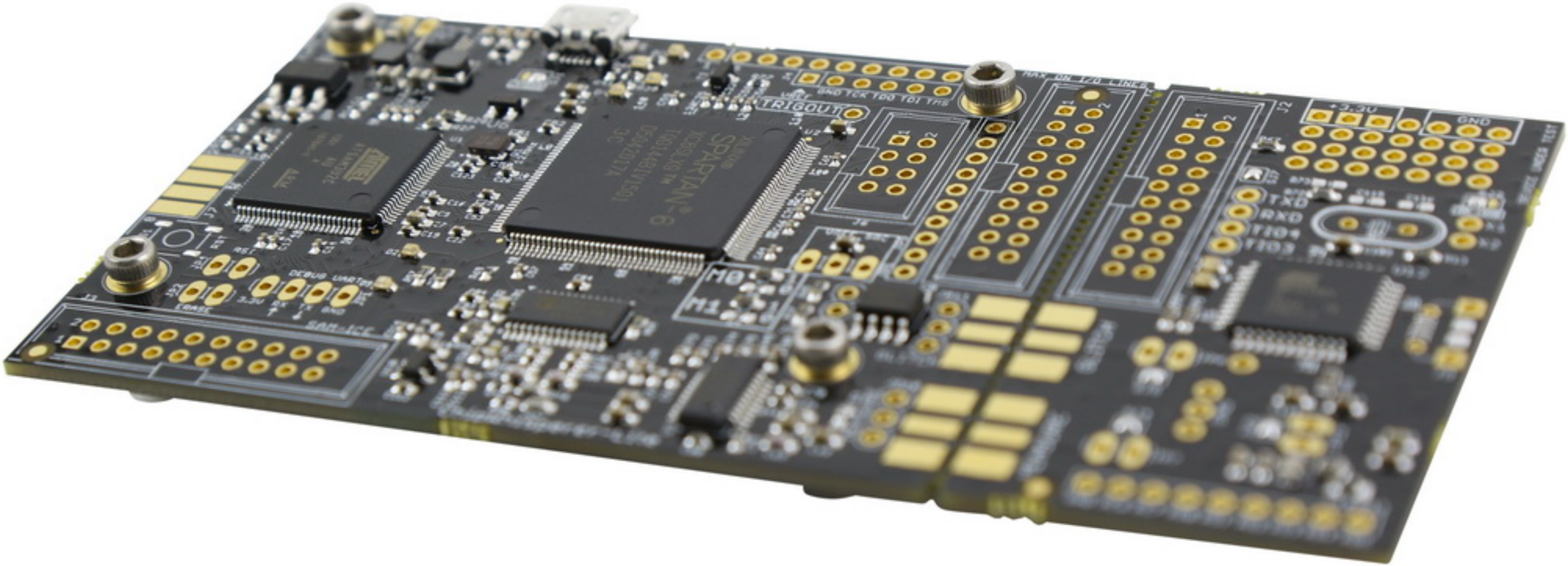
```
raw    = (char *)items;
length = N * sizeof(items[0]);

while (--length) {
    send_byte(raw++);
}
```

```
  ; [snip]
  ; compute length
  MUL  R1, R11, R12

loop:
  DEC  R1, R1 ; --length
  JZ   finish
  CALL send_byte
  INC  R2, R2 ; raw++
  JMP  loop

finish:
  NOP
```



TARGET: DMA CONTROLLERS



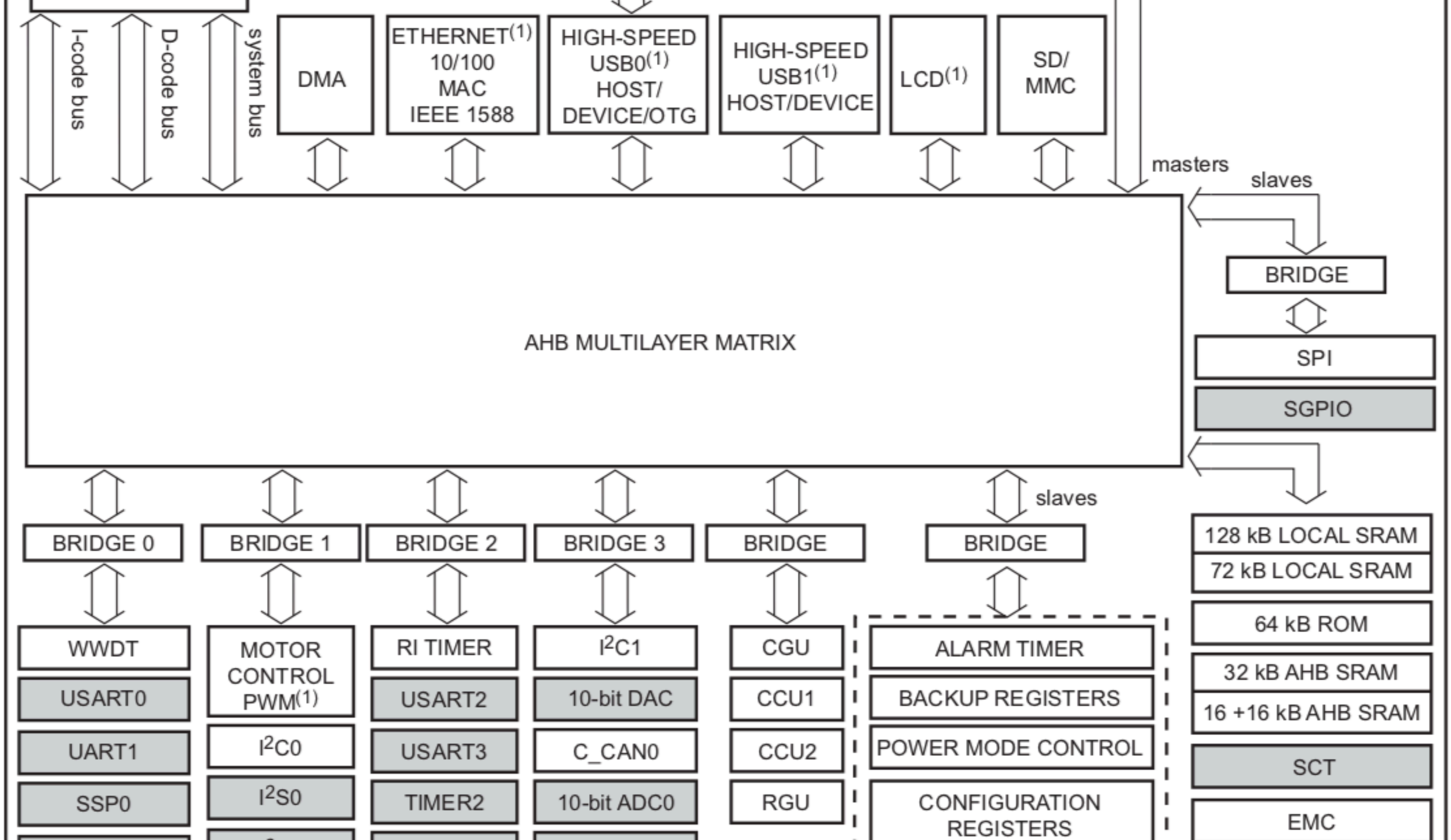


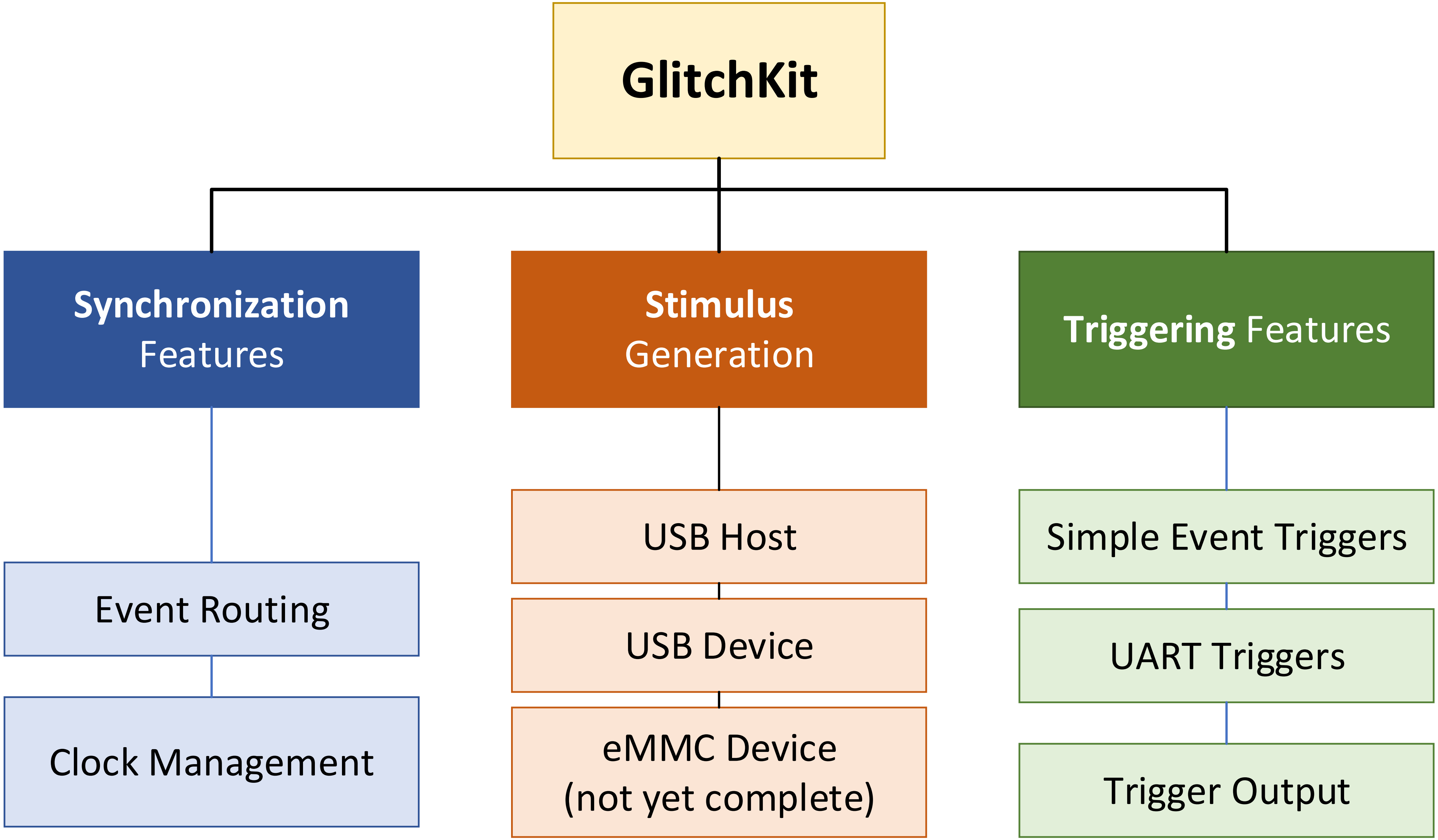
CHIPWHISPERER LITE GLITCHING & SIDE-CHANNEL BOARD

https://newae.com/tools/chipwhisperer/

https://github.com/newaetech/chipwhisperer

open source
hardware


I-code bus
D-code bus
system bus
DMA
ETHERNET(1)
10/100
MAC
IEEE 1588
HIGH-SPEED
USB0(1)
HOST/
DEVICE/OTG
HIGH-SPEED
USB1(1)
HOST/DEVICE
LCD(1)
SD/
MMC
masters
slaves
AHB MULTILAYER MATRIX
BRIDGE
SPI
SGPIO
slaves
BRIDGE 0
BRIDGE 1
BRIDGE 2
BRIDGE 3
BRIDGE
BRIDGE
128 kB LOCAL SRAM
72 kB LOCAL SRAM
64 kB ROM
32 kB AHB SRAM
16 +16 kB AHB SRAM
SCT
EMC
WWDT
USART0
UART1
SSP0
MOTOR
CONTROL
PWM(1)
I2C0
I2S0
RI TIMER
USART2
USART3
TIMER2
I2C1
10-bit DAC
C_CAN0
10-bit ADC0
CGU
CCU1
CCU2
RGU
ALARM TIMER
BACKUP REGISTERS
POWER MODE CONTROL
CONFIGURATION
REGISTERS



GlitchKit

Synchronization Features

Event Routing

Clock Management

Stimulus Generation

USB Host

USB Device

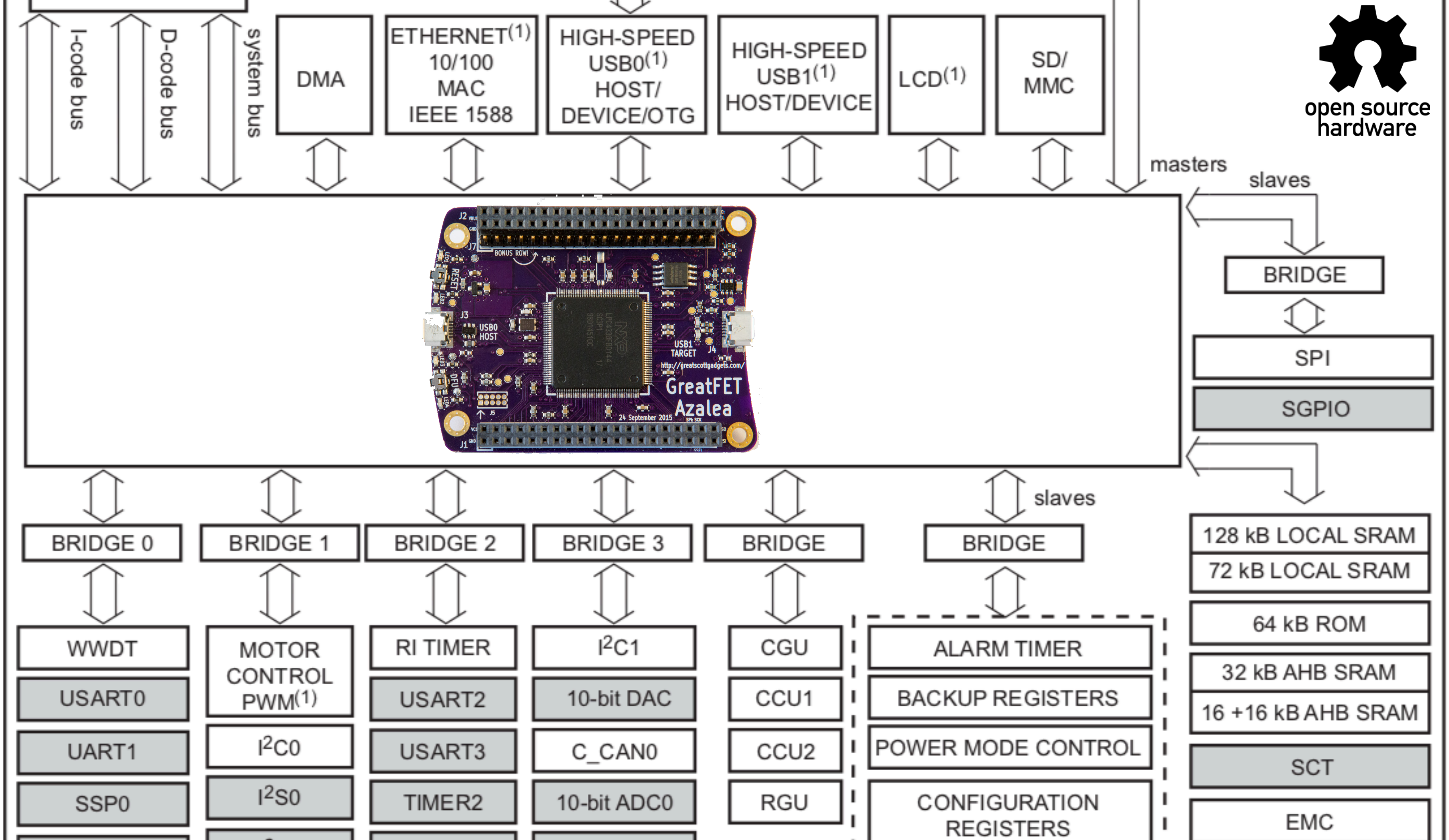
eMMC Device (not yet complete)

Triggering Features

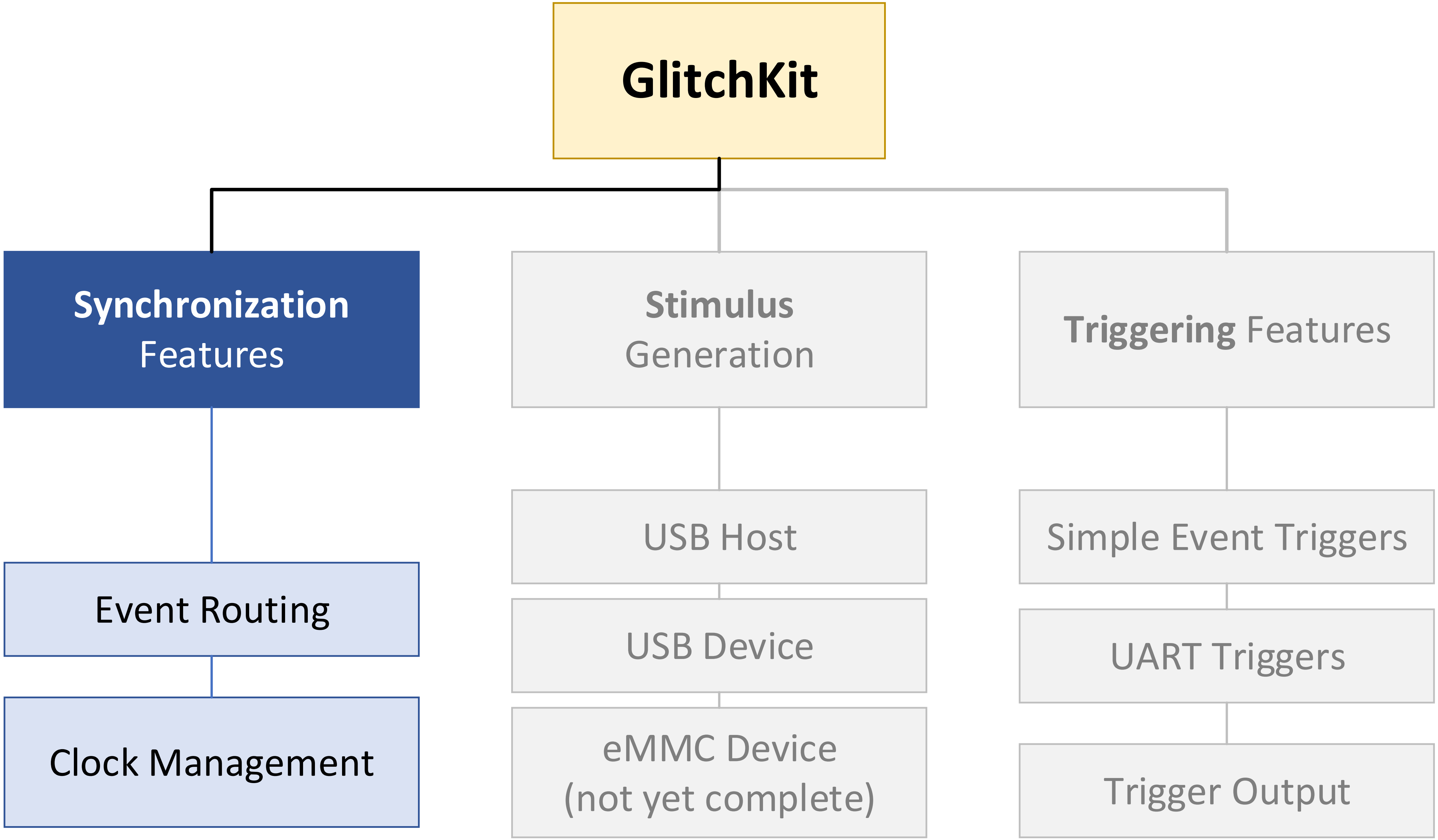
Simple Event Triggers

UART Triggers

Trigger Output



open source hardware



GlitchKit

Synchronization Features

Event Routing

Clock Management

Stimulus Generation

USB Host

USB Device

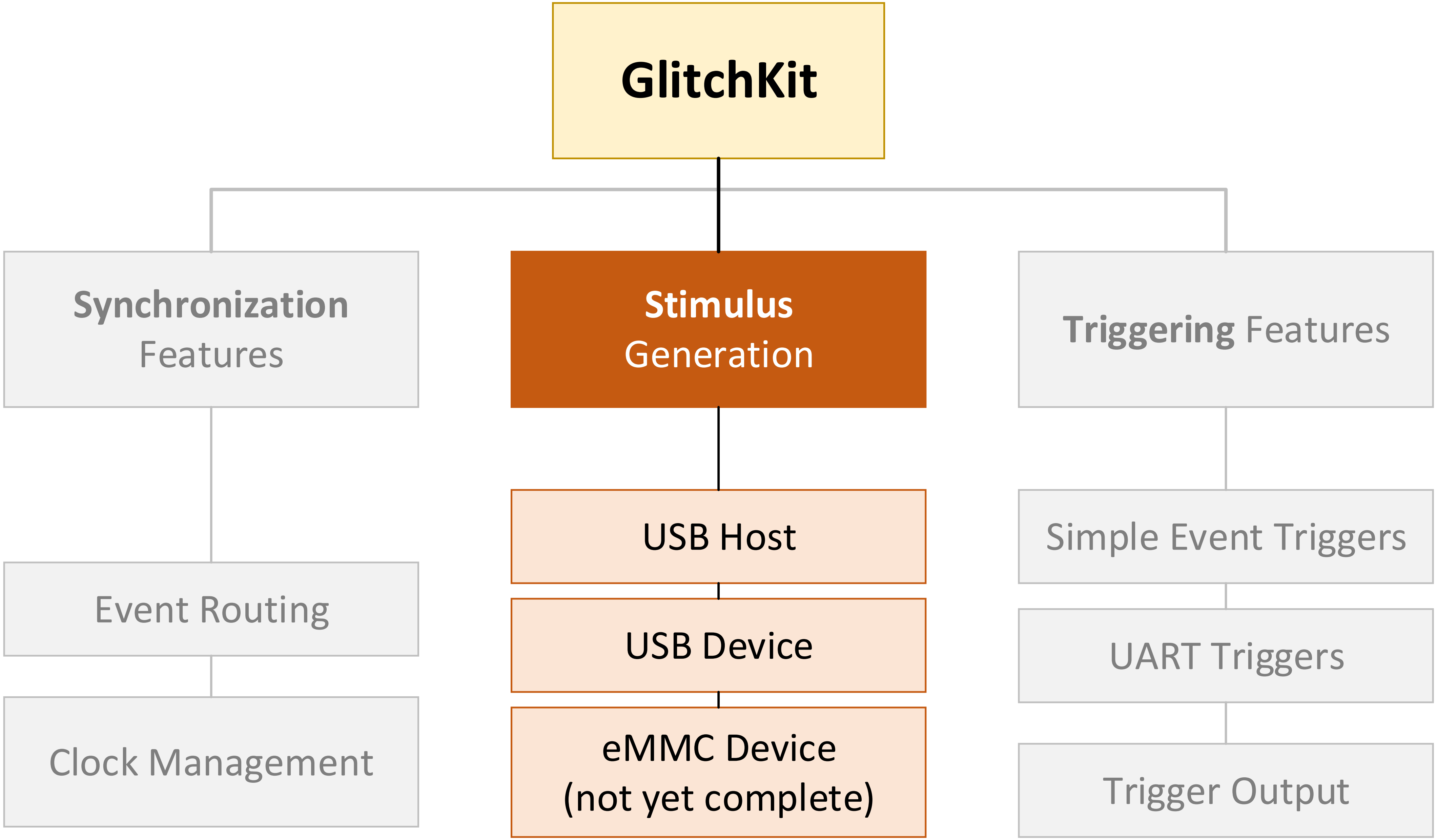
eMMC Device (not yet complete)

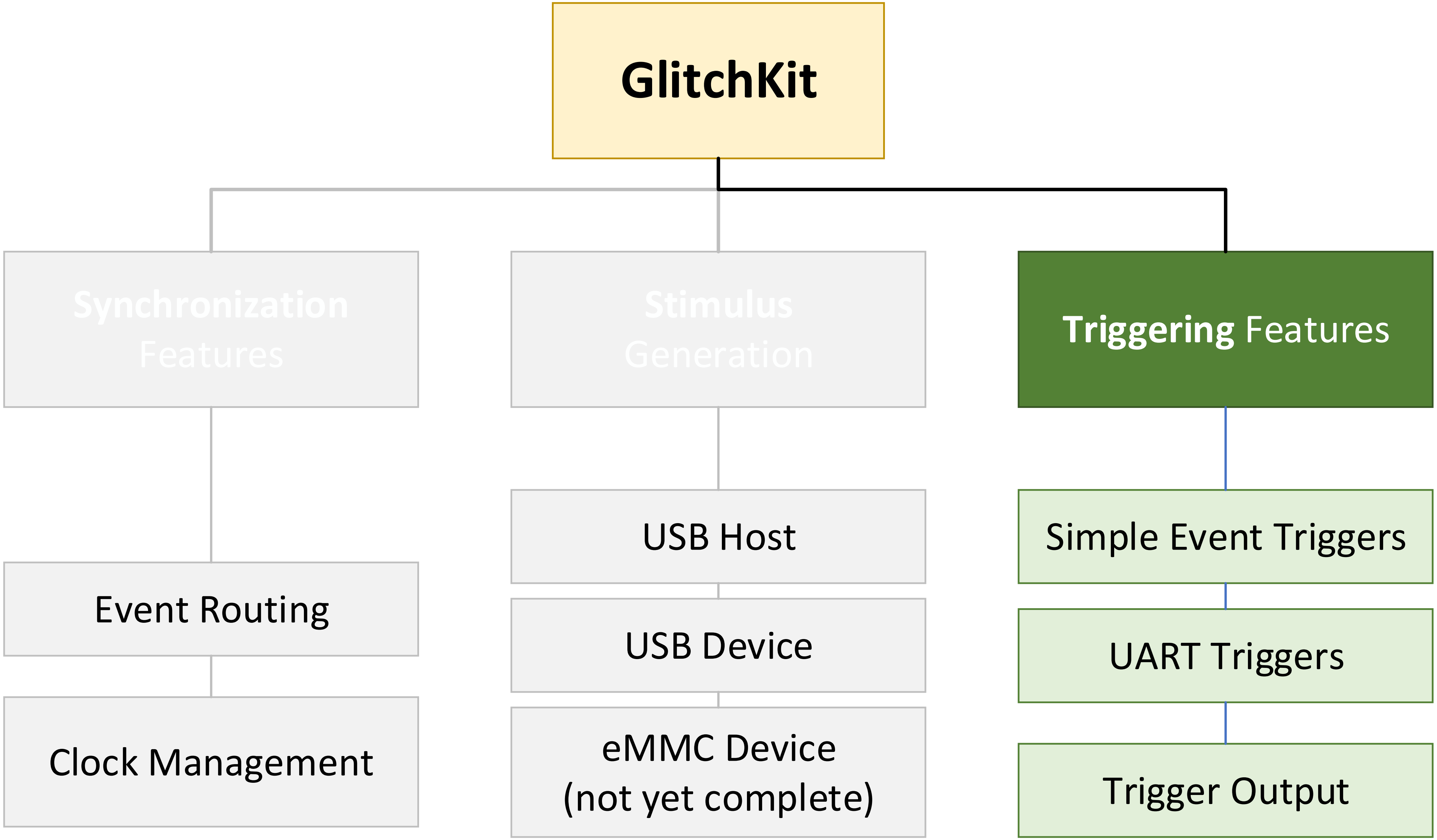
Triggering Features

Simple Event Triggers

UART Triggers

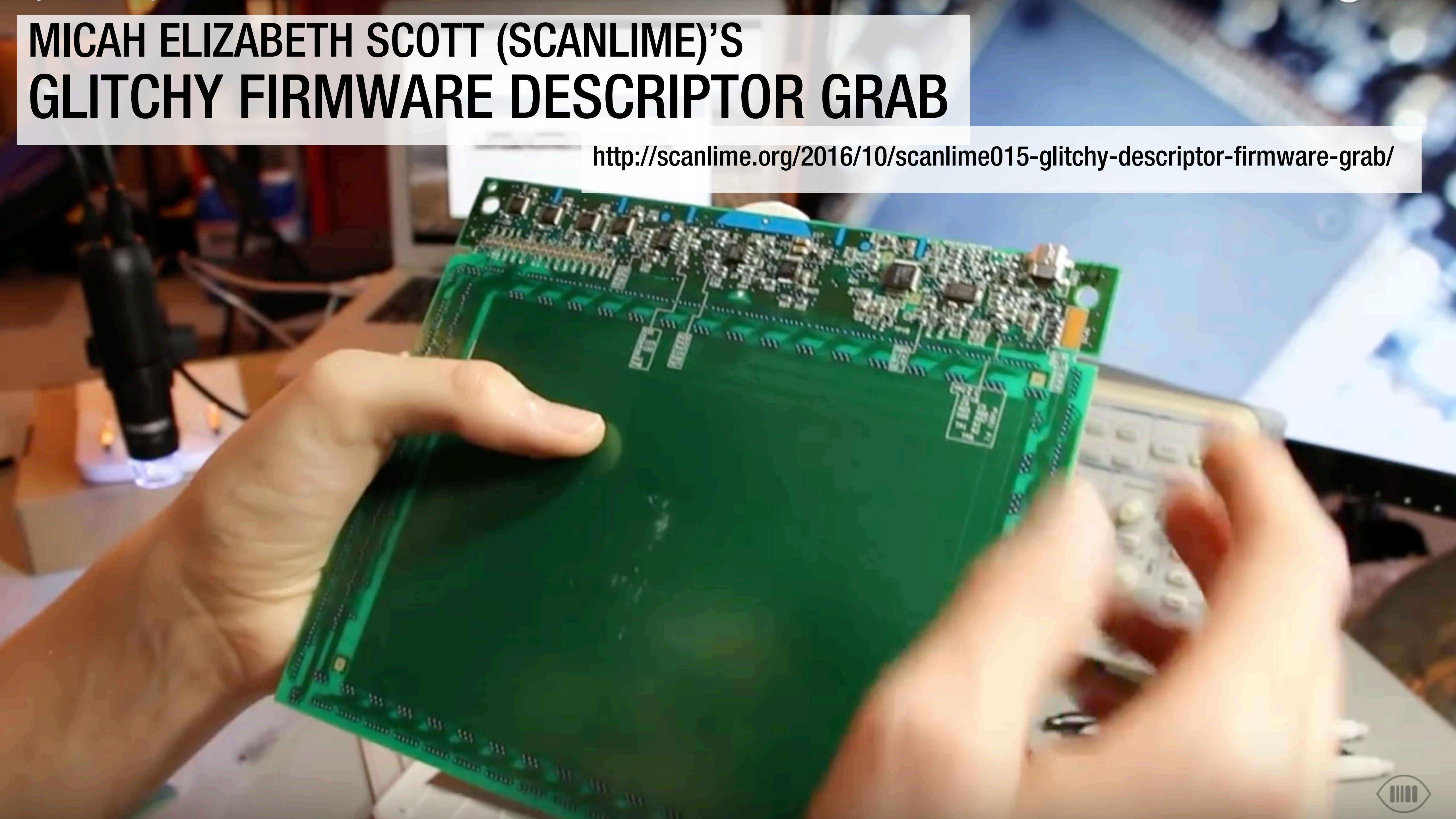
Trigger Output

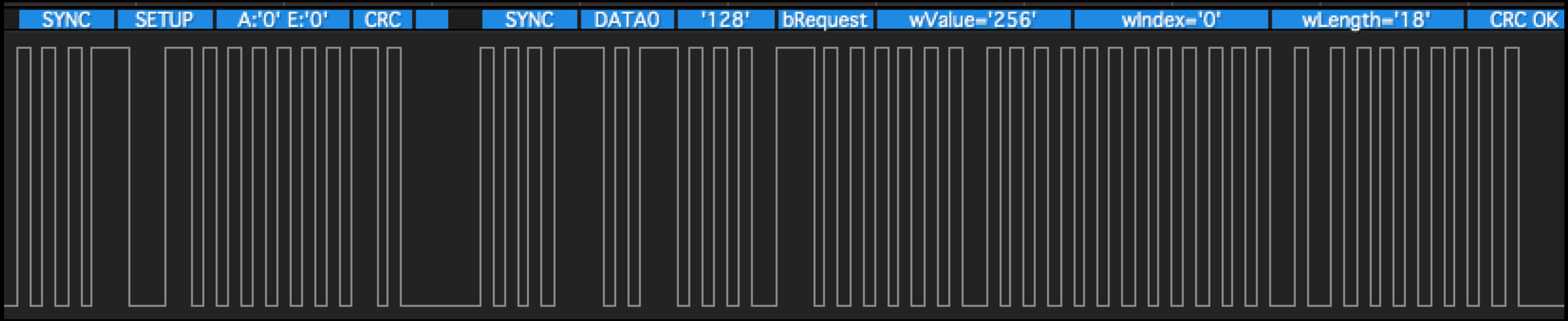

GlitchKit
Synchronization Features
Stimulus Generation
Triggering Features
Event Routing
Clock Management
USB Host
USB Device
eMMC Device (not yet complete)
Simple Event Triggers
UART Triggers
Trigger Output

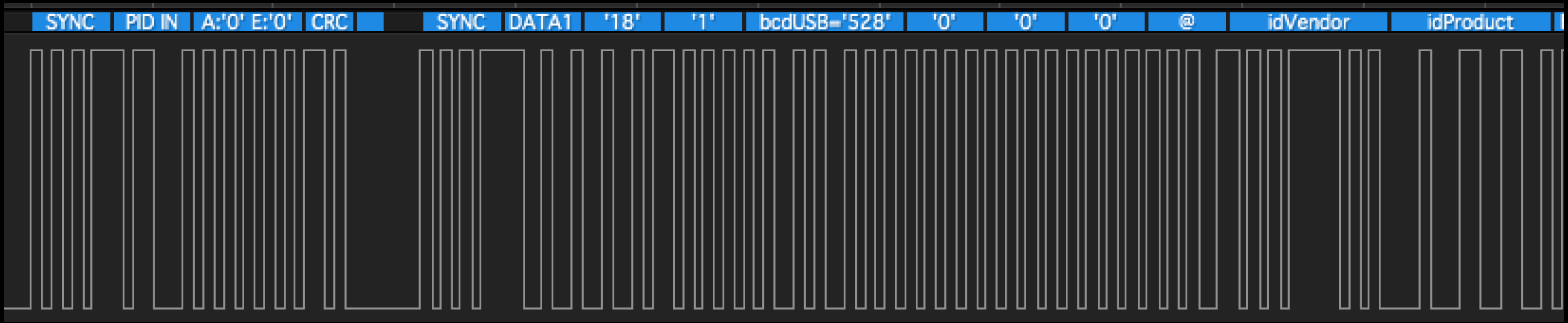

GlitchKit
Synchronization Features
Stimulus Generation
Triggering Features
Event Routing
Clock Management
USB Host
USB Device
eMMC Device (not yet complete)
Simple Event Triggers
UART Triggers
Trigger Output

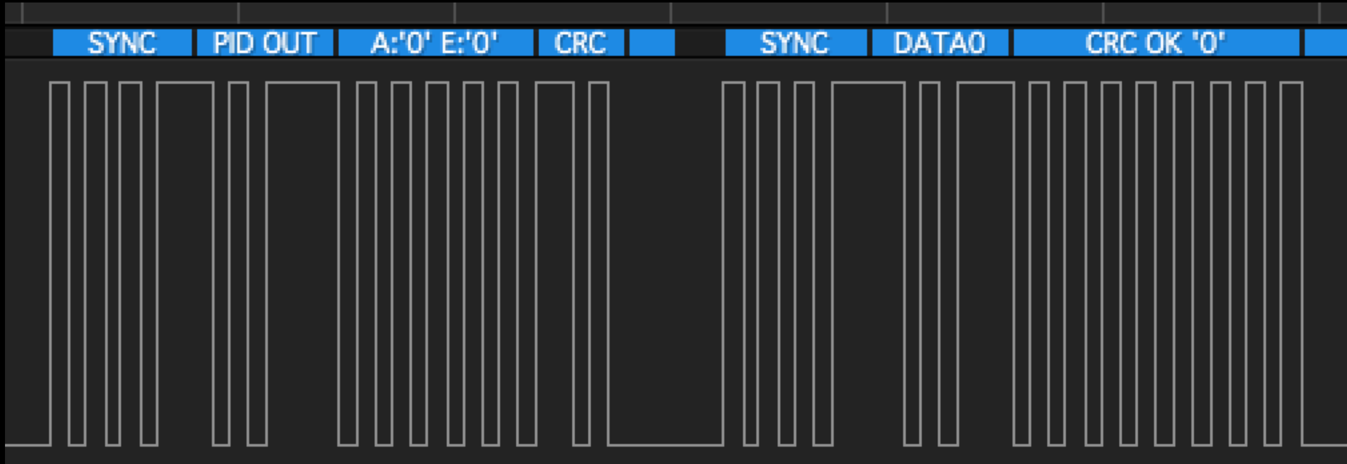
MICAH ELIZABETH SCOTT (SCANLIME)'S
GLITCHY FIRMWARE DESCRIPTOR GRAB

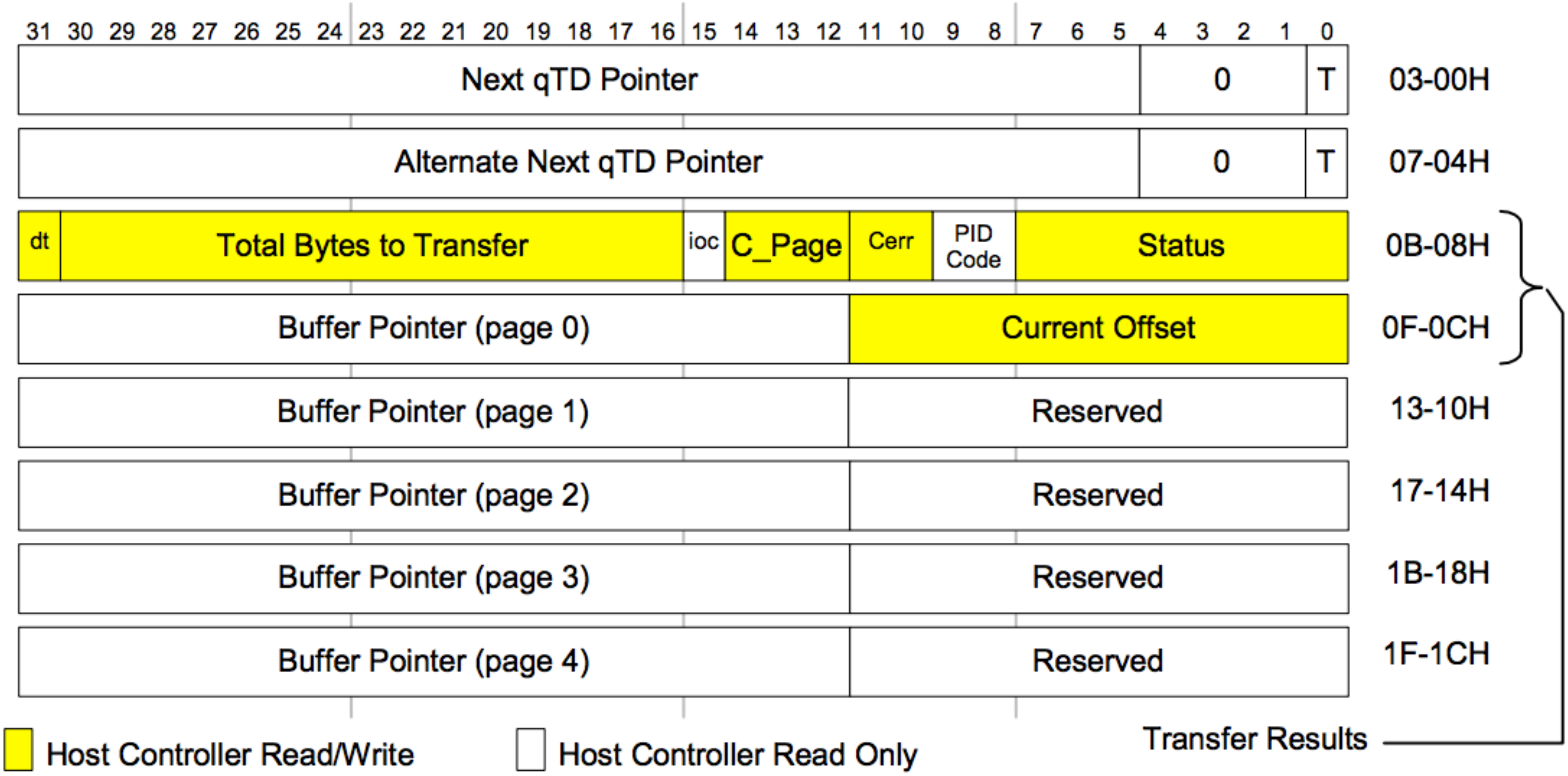
http://scanlime.org/2016/10/scanlime015-glitchy-descriptor-firmware-grab/




SYNC
SETUP
A:'0' E:'0'
CRC
SYNC
DATA0
'128'
bRequest
wValue='256'
wIndex='0'
wLength='18'
CRC OK


SYNC
PID IN
A:'0' E:'0'
CRC
SYNC
DATA1
'18'
'1'
bcdUSB='528'
'0'
'0'
'0'
@
idVendor
idProduct


SYNC
PID OUT
A:'0' E:'0'
CRC
SYNC
DATA0
CRC OK '0'



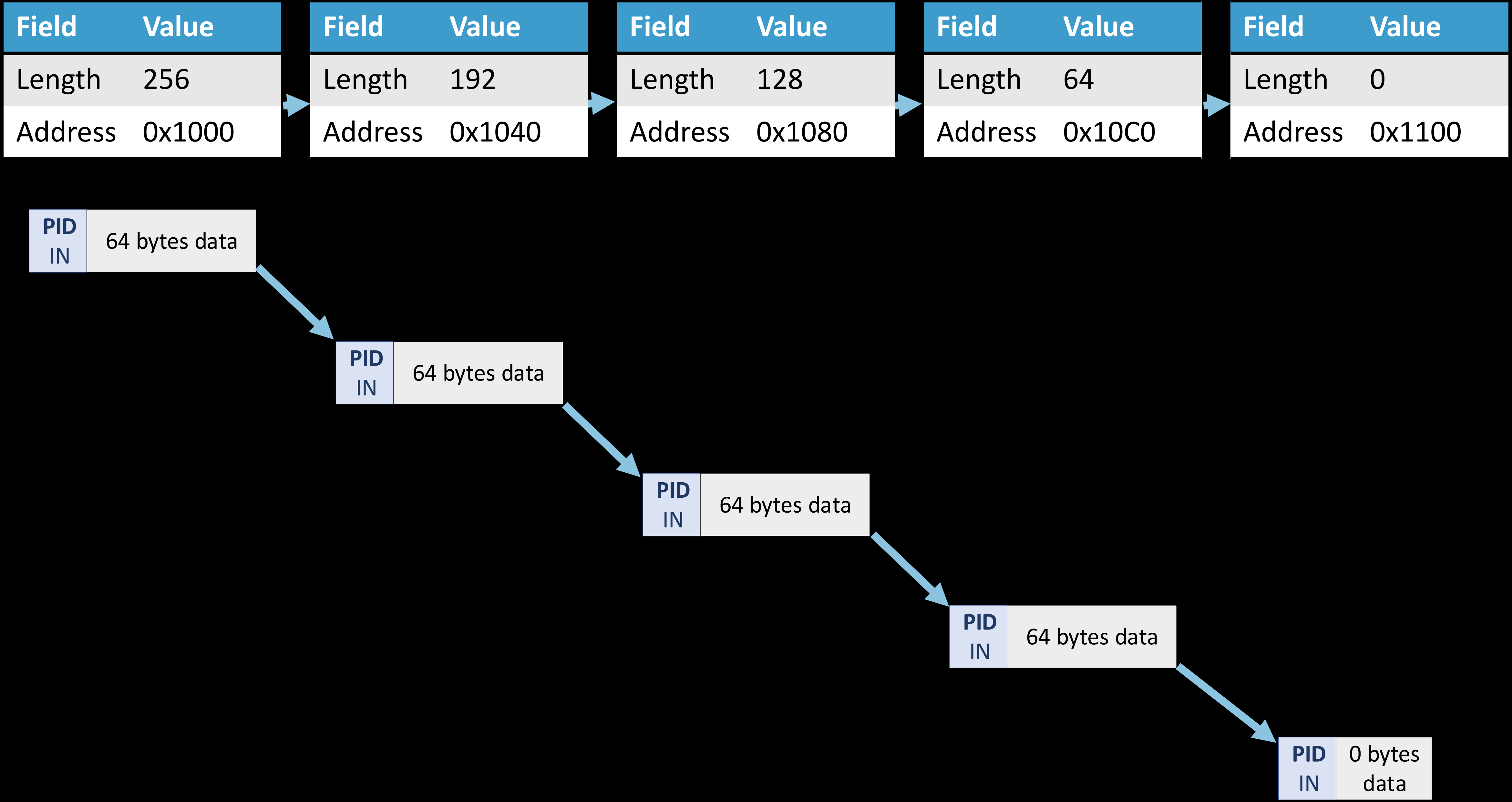
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Next qTD Pointer					0	T	03-00H
Alternate Next qTD Pointer					0	T	07-04H
dt	Total Bytes to Transfer	ioc	C_Page	Cerr	PID Code	Status	0B-08H
Buffer Pointer (page 0)				Current Offset			0F-0CH
Buffer Pointer (page 1)				Reserved			13-10H
Buffer Pointer (page 2)				Reserved			17-14H
Buffer Pointer (page 3)				Reserved			1B-18H
Buffer Pointer (page 4)				Reserved			1F-1CH

🟨 Host Controller Read/Write

□ Host Controller Read Only

Transfer Results



Field	Value
Length	256
Address	0x1000

Field	Value
Length	192
Address	0x1040

Field	Value
Length	128
Address	0x1080

Field	Value
Length	64
Address	0x10C0

Field	Value
Length	0
Address	0x1100

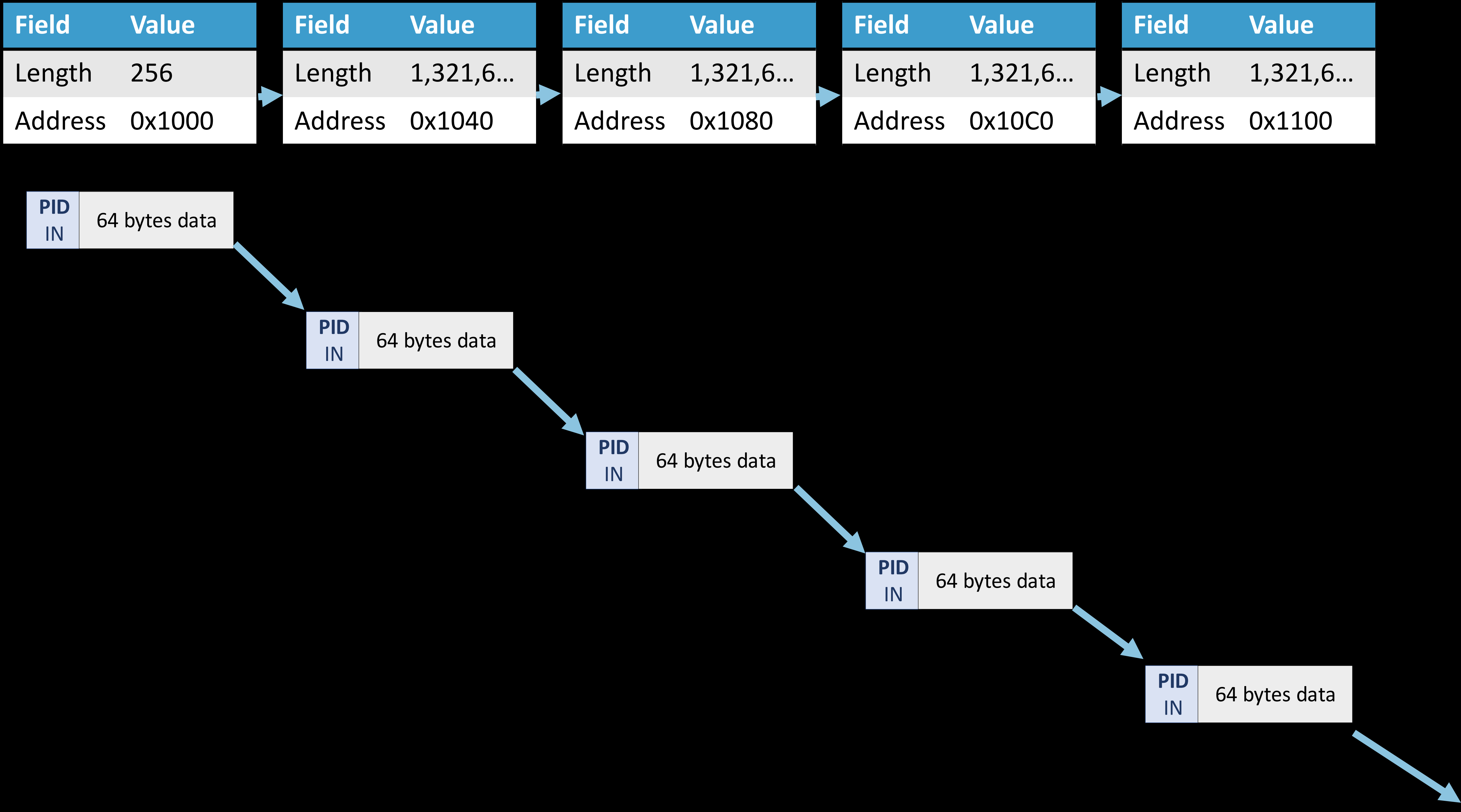
PID IN 64 bytes data

PID IN 64 bytes data

PID IN 64 bytes data

PID IN 64 bytes data

PID IN 0 bytes data



Field	Value
Length	256
Address	0x1000

Field	Value
Length	1,321,6…
Address	0x1040

Field	Value
Length	1,321,6…
Address	0x1080

Field	Value
Length	1,321,6…
Address	0x10C0

Field	Value
Length	1,321,6…
Address	0x1100

PID IN 64 bytes data

PID IN 64 bytes data

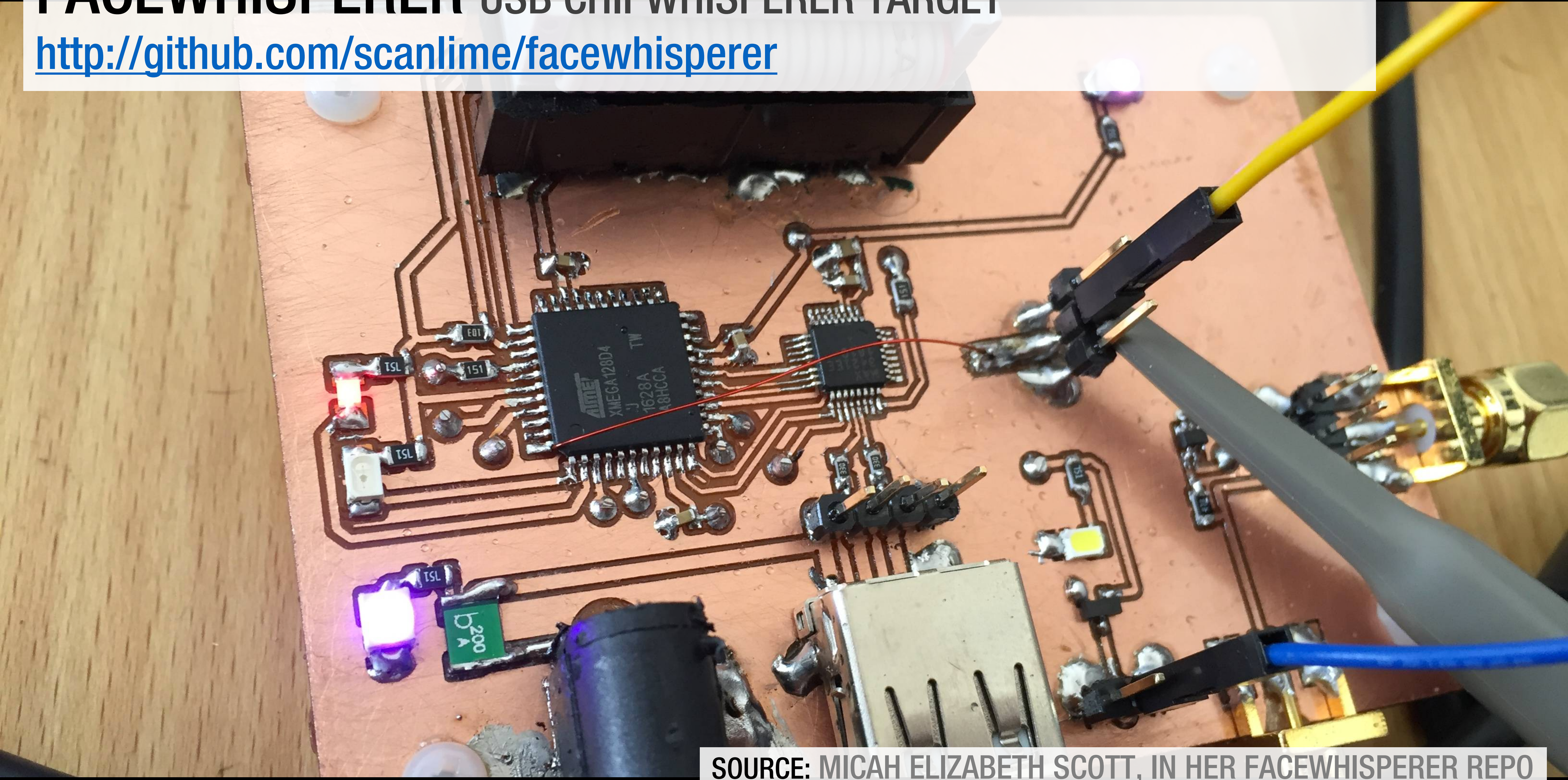
PID IN 64 bytes data

PID IN 64 bytes data

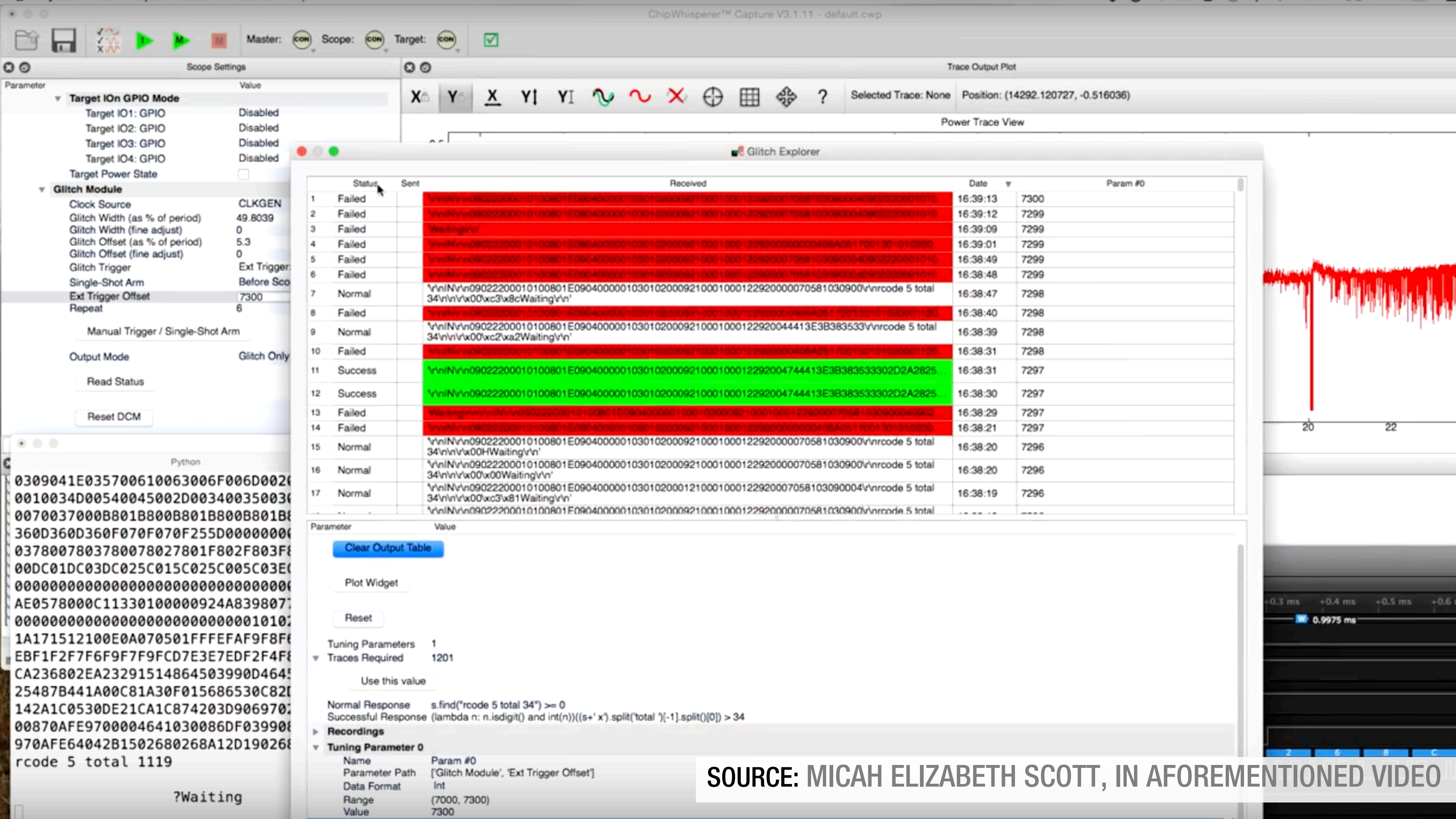
PID IN 64 bytes data

FACEWHISPERER USB CHIPWHISPERER TARGET

http://github.com/scanlime/facewhisperer



SOURCE: MICAH ELIZABETH SCOTT, IN HER FACEWHISPERER REPO



Parameter	Value
Target IOn GPIO Mode	
Target IO1: GPIO	Disabled
Target IO2: GPIO	Disabled
Target IO3: GPIO	Disabled
Target IO4: GPIO	Disabled
Target Power State	□
Glitch Module	
Clock Source	CLKGEN
Glitch Width (as % of period)	49.8039
Glitch Width (fine adjust)	0
Glitch Offset (as % of period)	5.3
Glitch Offset (fine adjust)	0
Glitch Trigger	Ext Trigger:
Single-Shot Arm	Before Sco
Ext Trigger Offset	7300
Repeat	6
Manual Trigger / Single-Shot Arm	
Output Mode	Glitch Only
Read Status	
Reset DCM	

	Status	Sent	Received	Date	Param #0
1	Failed		[illegible]	16:39:13	7300
2	Failed		[illegible]	16:39:12	7299
3	Failed		[illegible]	16:39:09	7299
4	Failed		[illegible]	16:39:01	7299
5	Failed		[illegible]	16:38:49	7299
6	Failed		[illegible]	16:38:48	7299
7	Normal		'\r\nIN\r\n0902220001010080 1E0904000001030102000921000100012292000007058103090 0\r\nrcode 5 total 34\n\n\r\x00\xc3\x8cWaiting\r\n'	16:38:47	7298
8	Failed		[illegible]	16:38:40	7298
9	Normal		'\r\nIN\r\n0902220001010080 1E09040000010301020009210001000122920044413E3B383533\r\nrcode 5 total 34\n\n\r\x00\xc2\xa2Waiting\r\n'	16:38:39	7298
10	Failed		[illegible]	16:38:31	7298
11	Success		'\r\nIN\r\n0902220001010080 1E090400000103010200092100010001229200474441 3E3B383533302D2A2825...	16:38:31	7297
12	Success		'\r\nIN\r\n0902220001010080 1E090400000103010200092100010001229200474441 3E3B383533302D2A2825...	16:38:30	7297
13	Failed		[illegible]	16:38:29	7297
14	Failed		[illegible]	16:38:21	7297
15	Normal		'\r\nIN\r\n0902220001010080 1E0904000001030102000921000100012292000007058103090 0\r\nrcode 5 total 34\n\n\r\x00HWaiting\r\n'	16:38:20	7296
16	Normal		'\r\nIN\r\n0902220001010080 1E0904000001030102000921000100012292000007058103090 0\r\nrcode 5 total 34\n\n\r\x00\x00Waiting\r\n'	16:38:20	7296
17	Normal		'\r\nIN\r\n0902220001010080 1E09040000010301020001210001000122920007058103090004\r\nrcode 5 total 34\n\n\r\x00\xc3\x81Waiting\r\n'	16:38:19	7296

Parameter	Value
Clear Output Table	
Plot Widget	
Reset	
Tuning Parameters	1
Traces Required	1201
Use this value	
Normal Response	s.find("rcode 5 total 34") >= 0
Successful Response	(lambda n: n.isdigit() and int(n))((s+' x').split('total ')[-1].split()[0]) > 34
Recordings	
Tuning Parameter 0	
Name	Param #0
Parameter Path	['Glitch Module', 'Ext Trigger Offset']
Data Format	Int
Range	(7000, 7300)
Value	7300

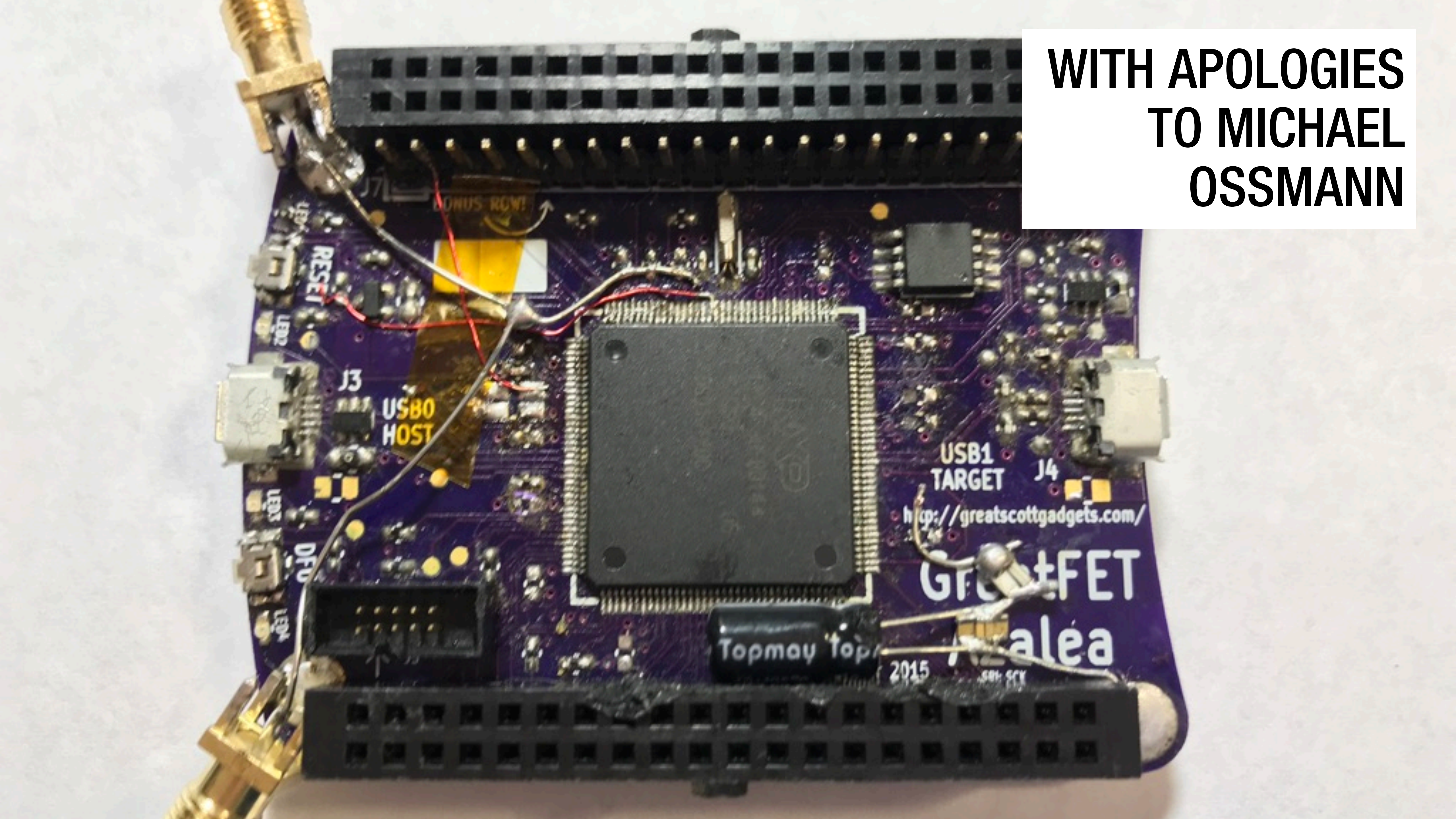
```
0309041E03570061006300 6F006D002(
0010034D00540045002D0034003500 3(
0070037000B801B800B801B800B801B8
360D360D360F070F070F255D0000000(
03780078037800780278 01F802F803F8
00DC01DC03DC025C015C025C005C03E(
000000000000000000000000000000(
AE0578000C11330100000924A839807
0000000000000000000000000001010
1A171512100E0A070501FFFEFAF9F8F(
EBF1F2F7F6F9F7F9FCD7E3E7EDF2F4F8
CA236802EA23291514864503990D464
25487B441A00C81A30F015686530C82I
142A1C0530DE21CA1C874203D906970
00870AFE970000464103008 6DF039908
970AFE64042B1502680268A12D19026
rcode 5 total 1119

              ?Waiting
```

SOURCE: MICAH ELIZABETH SCOTT, IN AFOREMENTIONED VIDEO

EQUIVALENT GLITCHKIT CODE

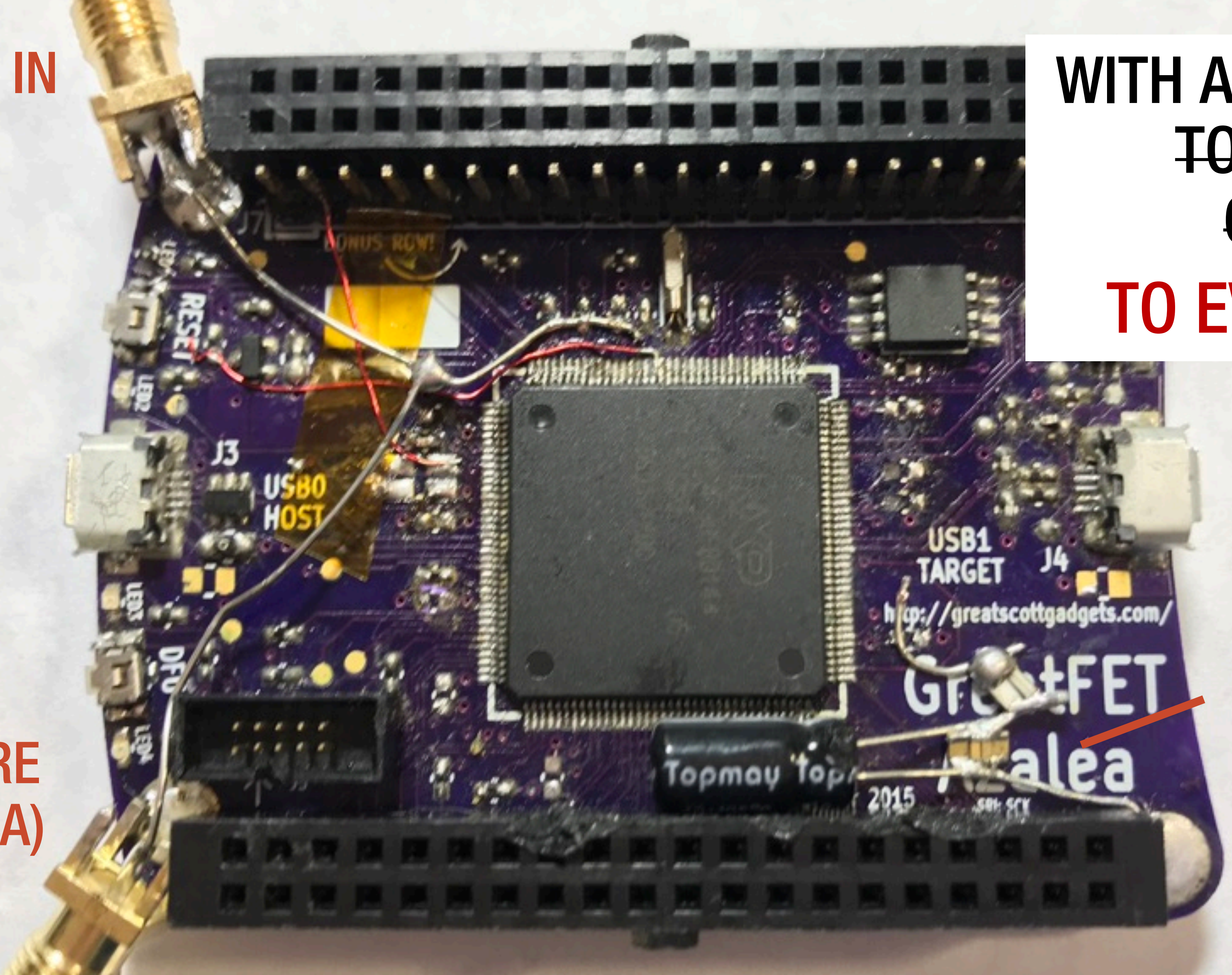
```
gf = GreatFET()
gf.switch_to_external_clock()
gf.glitchkit.provide_target_clock(VBUS_ENABLED);

gf.glitchkit.simple.watch_for_event(
    1, [('EDGE_RISING', 'J1_P7')])
gf.glitchkit.use_events_for_synchronization(COUNT_REACHED)

gf.glitchkit.trigger_on_events(HOST_SETUP_TRANSFER_QUEUED)
gf.glitchkit.usb.capture_control_in(request=GET_DESCRIPTOR,
    value=GET_DEVICE_DESCRIPTOR, length=18)
```

WITH APOLOGIES TO MICHAEL OSSMANN



WITH APOLOGIES ~~TO MICHAEL OSSMANN~~

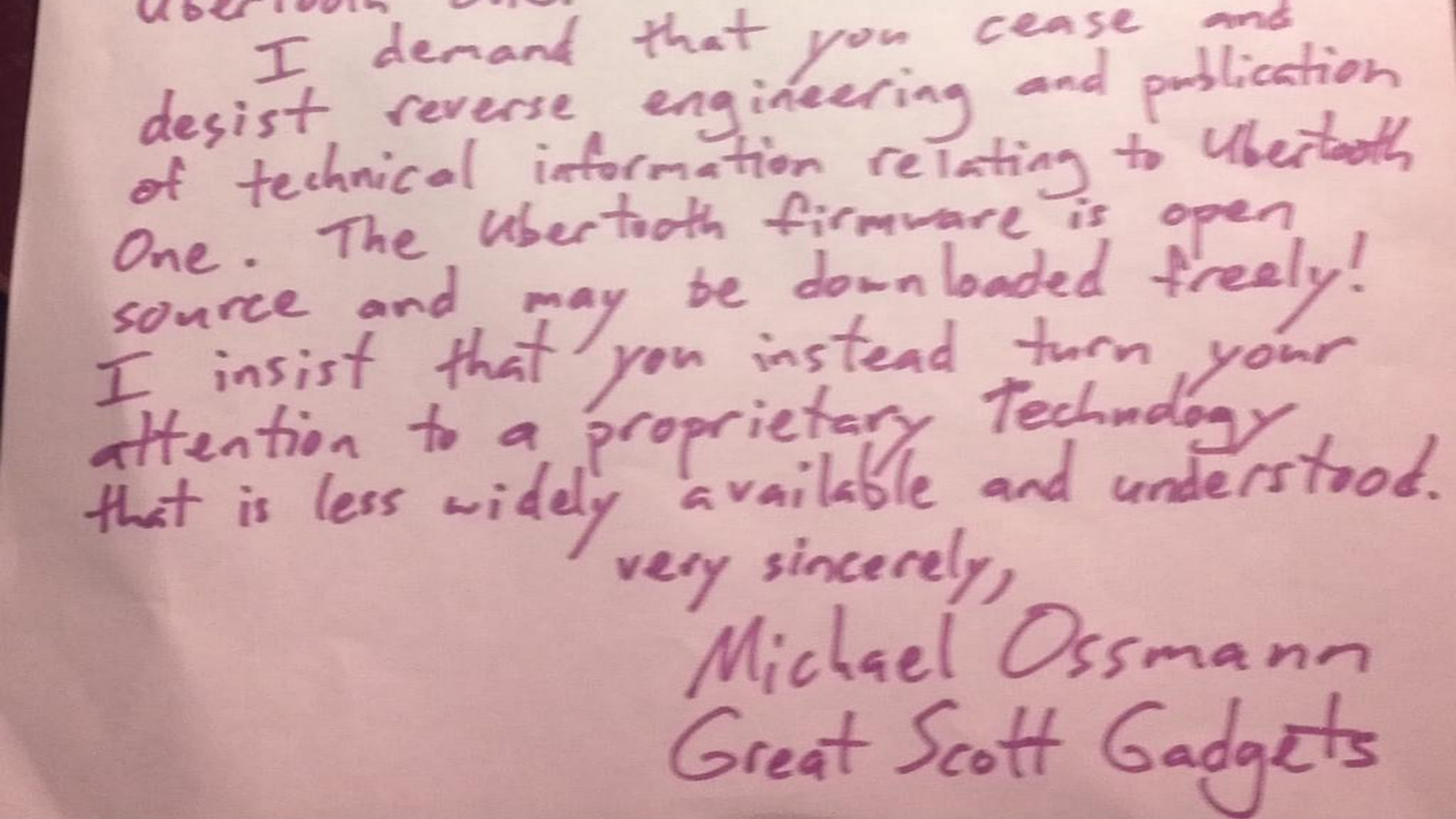
TO EVERYONE



GLITCH IN

HIGHER-Z DECOUPLING NETWORK

MEASURE OUT (SCA)



Ubertooth One

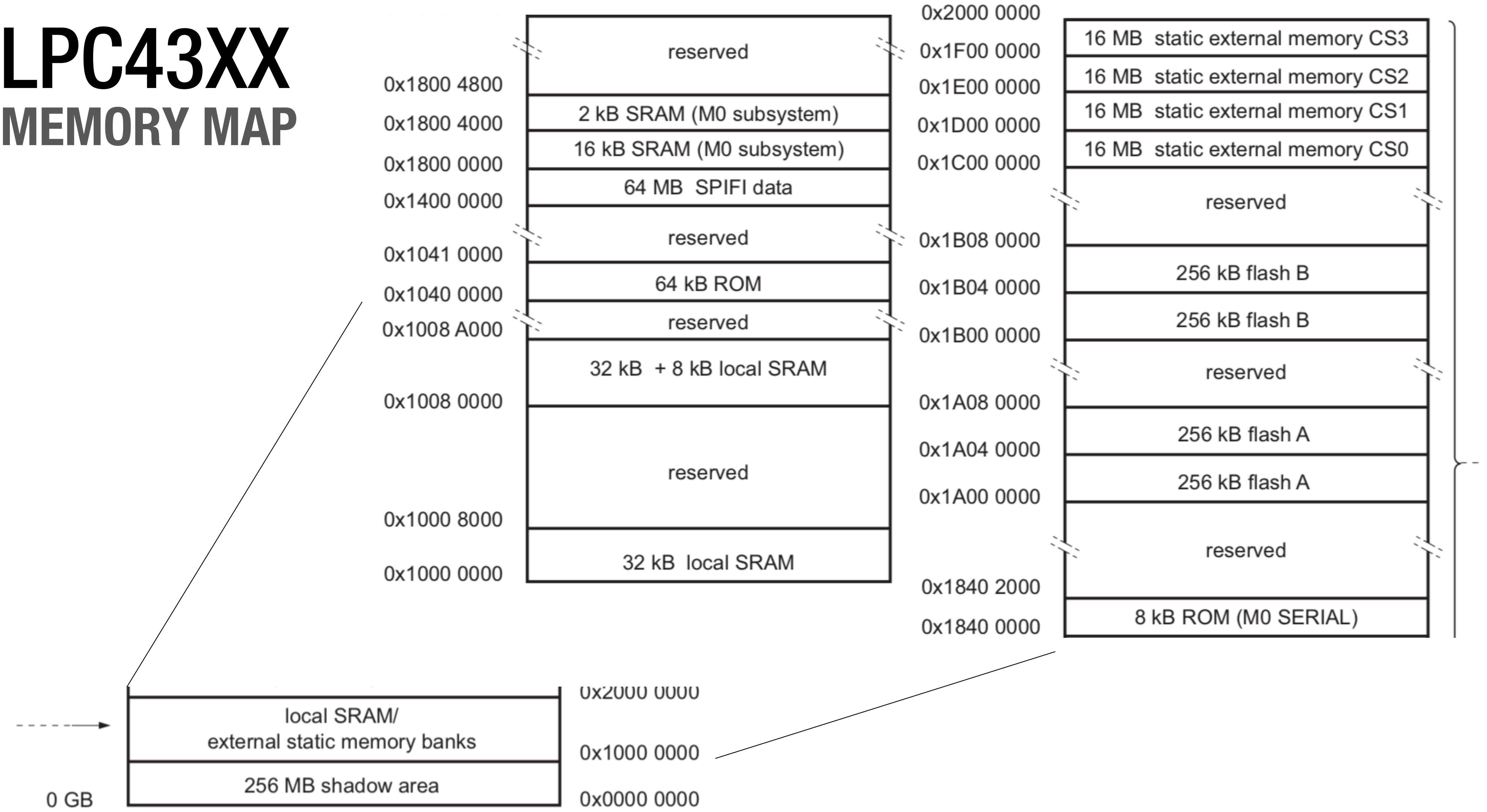
I demand that you cease and desist reverse engineering and publication of technical information relating to Ubertooth One. The Ubertooth firmware is open source and may be downloaded freely! I insist that you instead turn your attention to a proprietary Technology that is less widely available and understood.

very sincerely,

Michael Ossmann
Great Scott Gadgets

LPC43XX

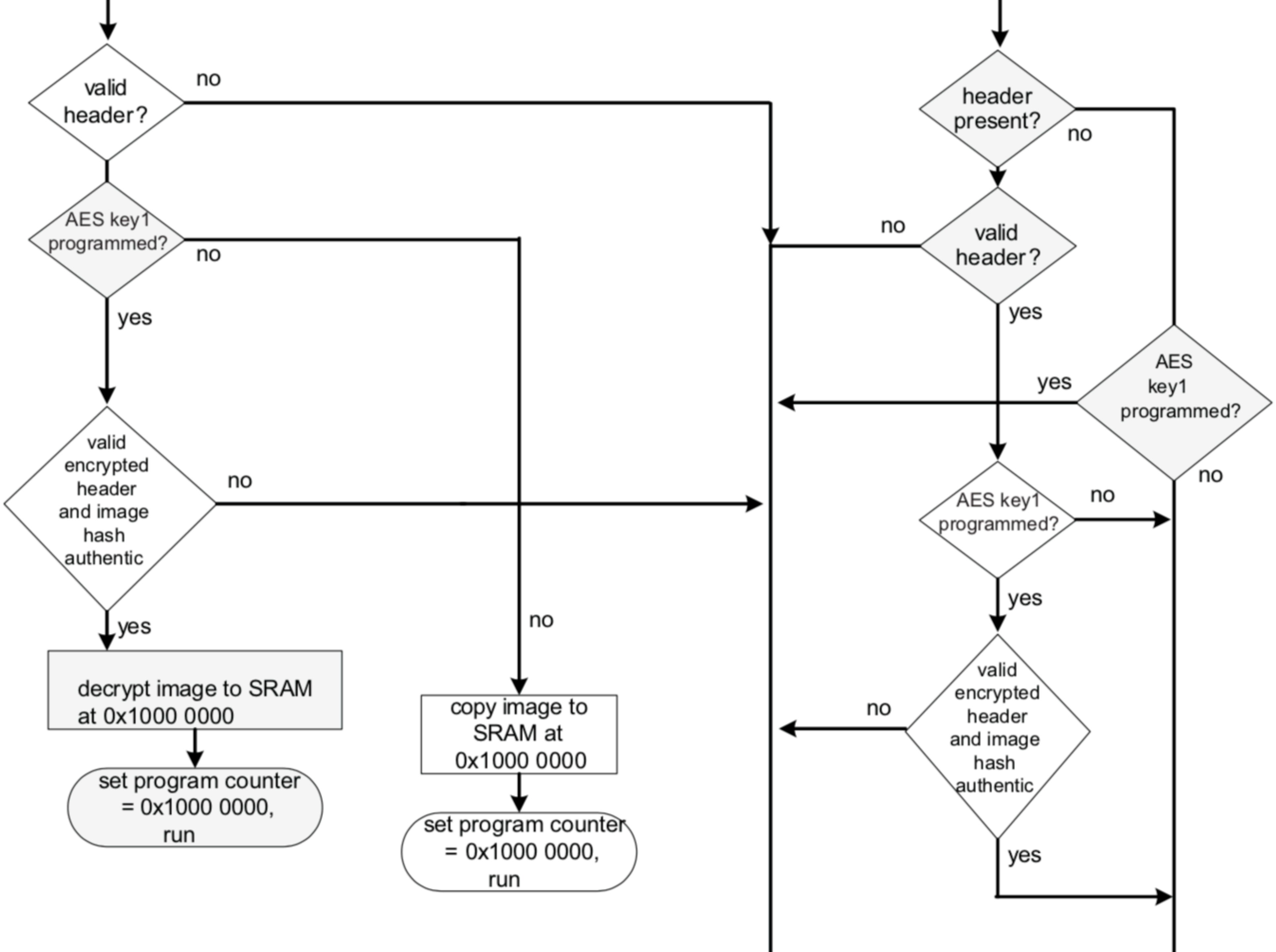
MEMORY MAP

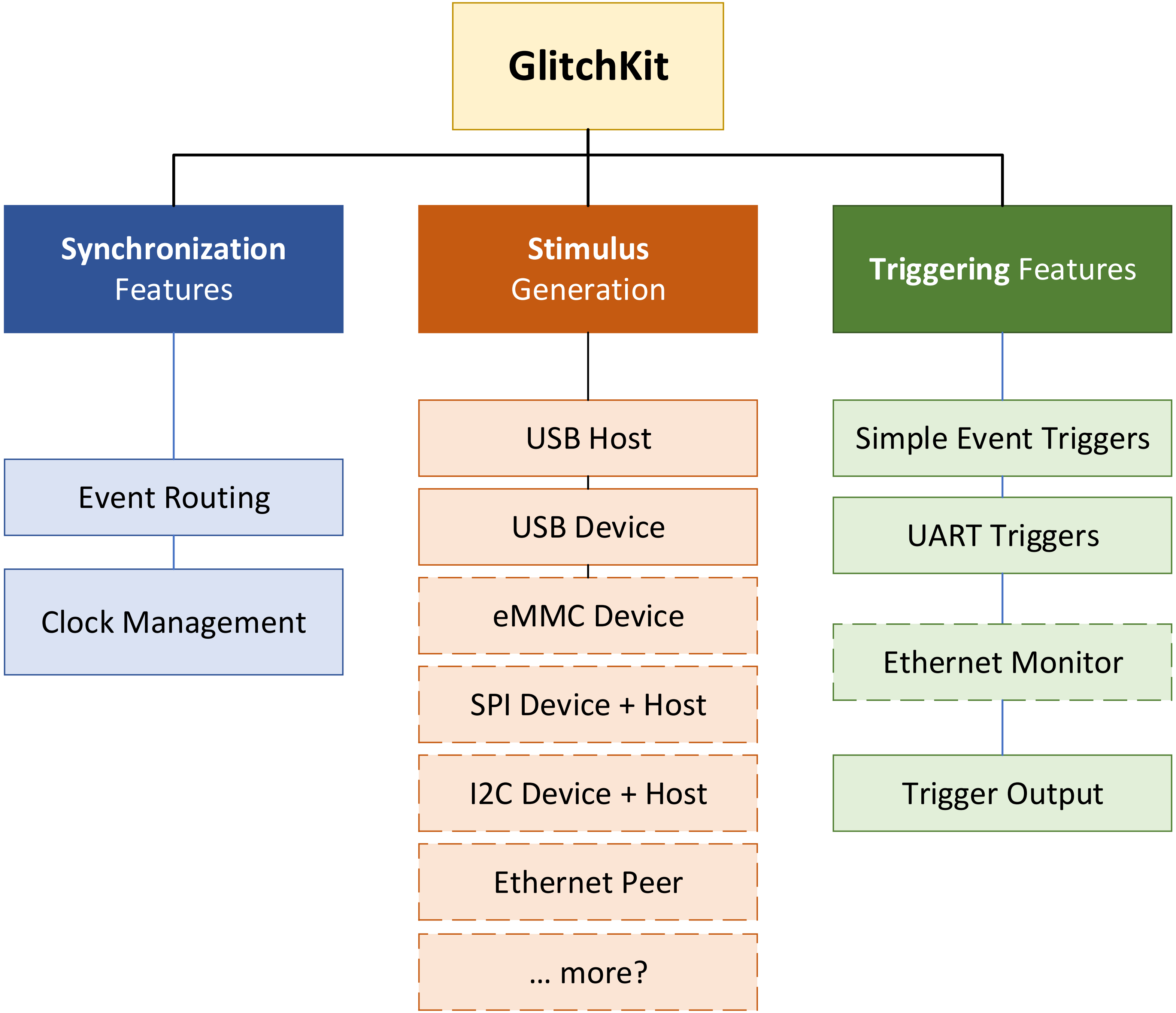


reserved
0x1800 4800
2 kB SRAM (M0 subsystem)
0x1800 4000
16 kB SRAM (M0 subsystem)
0x1800 0000
64 MB SPIFI data
0x1400 0000
reserved
0x1041 0000
64 kB ROM
0x1040 0000
reserved
0x1008 A000
32 kB + 8 kB local SRAM
0x1008 0000
reserved
0x1000 8000
32 kB local SRAM
0x1000 0000

0x2000 0000
16 MB static external memory CS3
0x1F00 0000
16 MB static external memory CS2
0x1E00 0000
16 MB static external memory CS1
0x1D00 0000
16 MB static external memory CS0
0x1C00 0000
reserved
0x1B08 0000
256 kB flash B
0x1B04 0000
256 kB flash B
0x1B00 0000
reserved
0x1A08 0000
256 kB flash A
0x1A04 0000
256 kB flash A
0x1A00 0000
reserved
0x1840 2000
8 kB ROM (M0 SERIAL)
0x1840 0000

0x2000 0000
local SRAM/
external static memory banks
0x1000 0000
256 MB shadow area
0 GB
0x0000 0000


valid header?
no
AES key1 programmed?
no
yes
valid encrypted header and image hash authentic
no
yes
decrypt image to SRAM at 0x1000 0000
set program counter = 0x1000 0000, run
no
copy image to SRAM at 0x1000 0000
set program counter = 0x1000 0000, run
header present?
no
no
valid header?
yes
yes
AES key1 programmed?
no
AES key1 programmed?
no
yes
no
valid encrypted header and image hash authentic
yes


GlitchKit

Synchronization Features

Event Routing

Clock Management

Stimulus Generation

USB Host

USB Device

eMMC Device

SPI Device + Host

I2C Device + Host

Ethernet Peer

… more?

Triggering Features

Simple Event Triggers

UART Triggers

Ethernet Monitor

Trigger Output

QUESTIONS?

THANKS FOR LISTENING!

JOIN US: https://github.com/glitchkit