Behind the Scenes of the

Badge

by Joe Grand
aka Kingpin

Me.

electrical engineer.

hardware hacker.

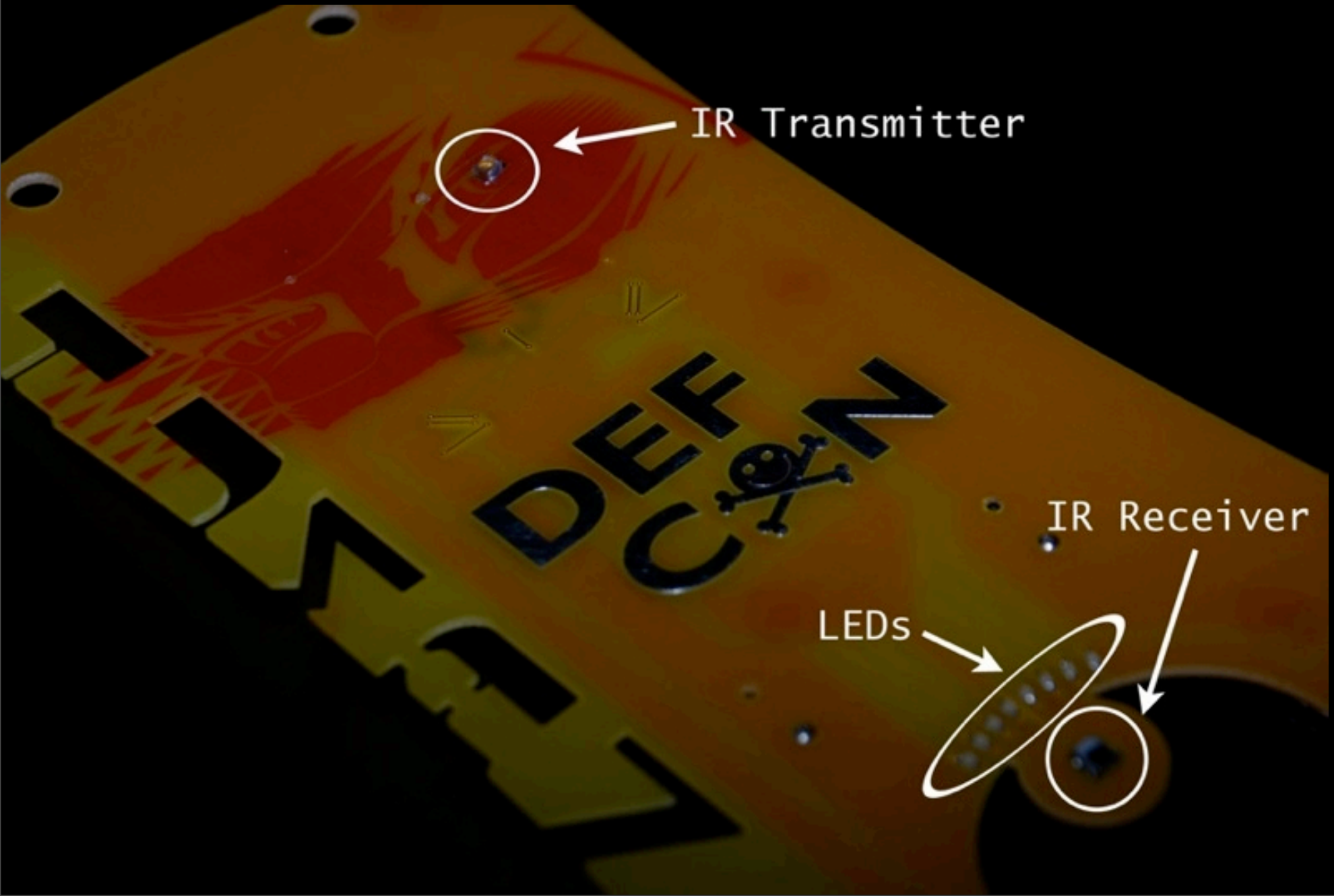
product designer.

Retrospective: DEFCON 14

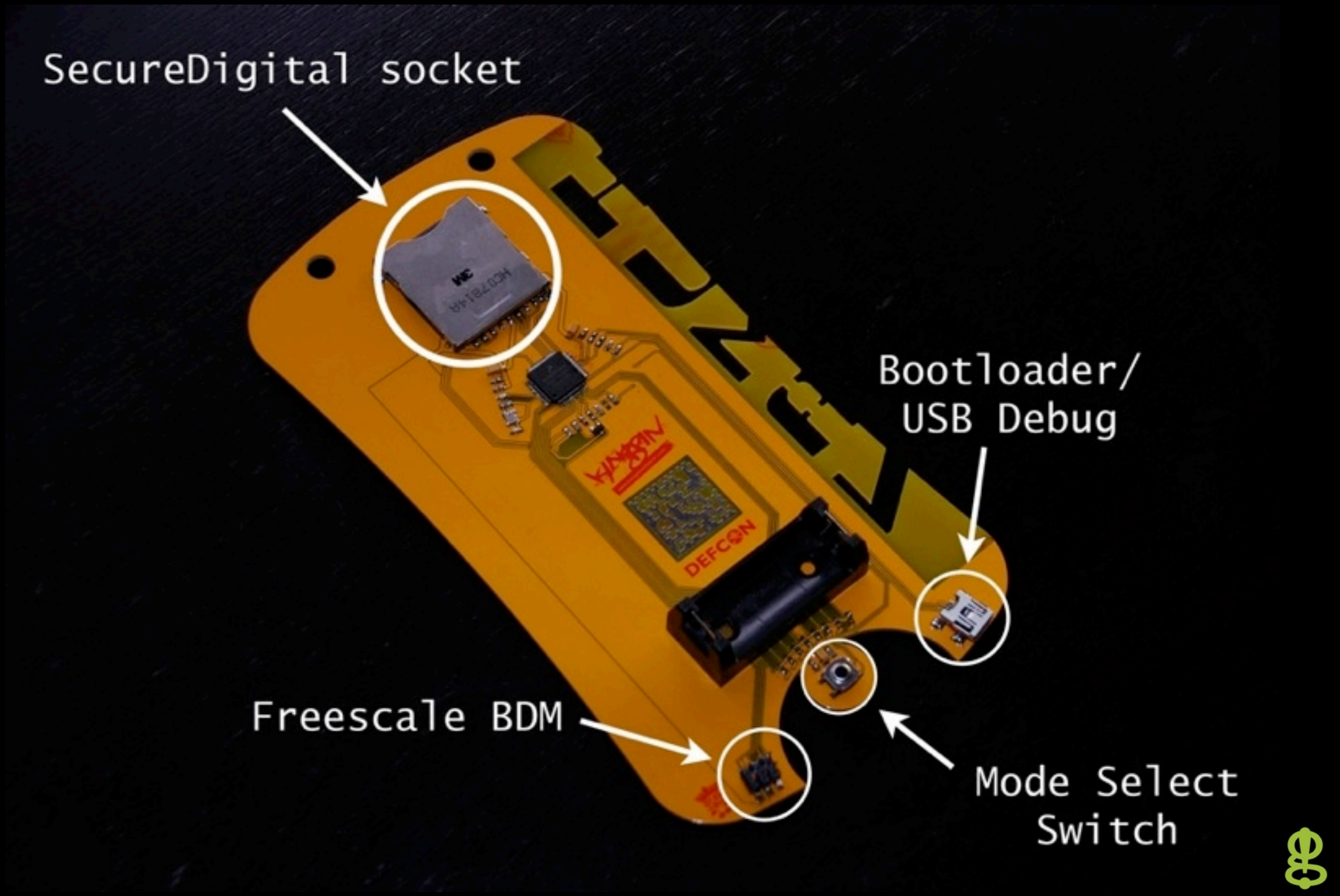
Retrospective: DEFCON 15

Retrospective: DEFCON 15

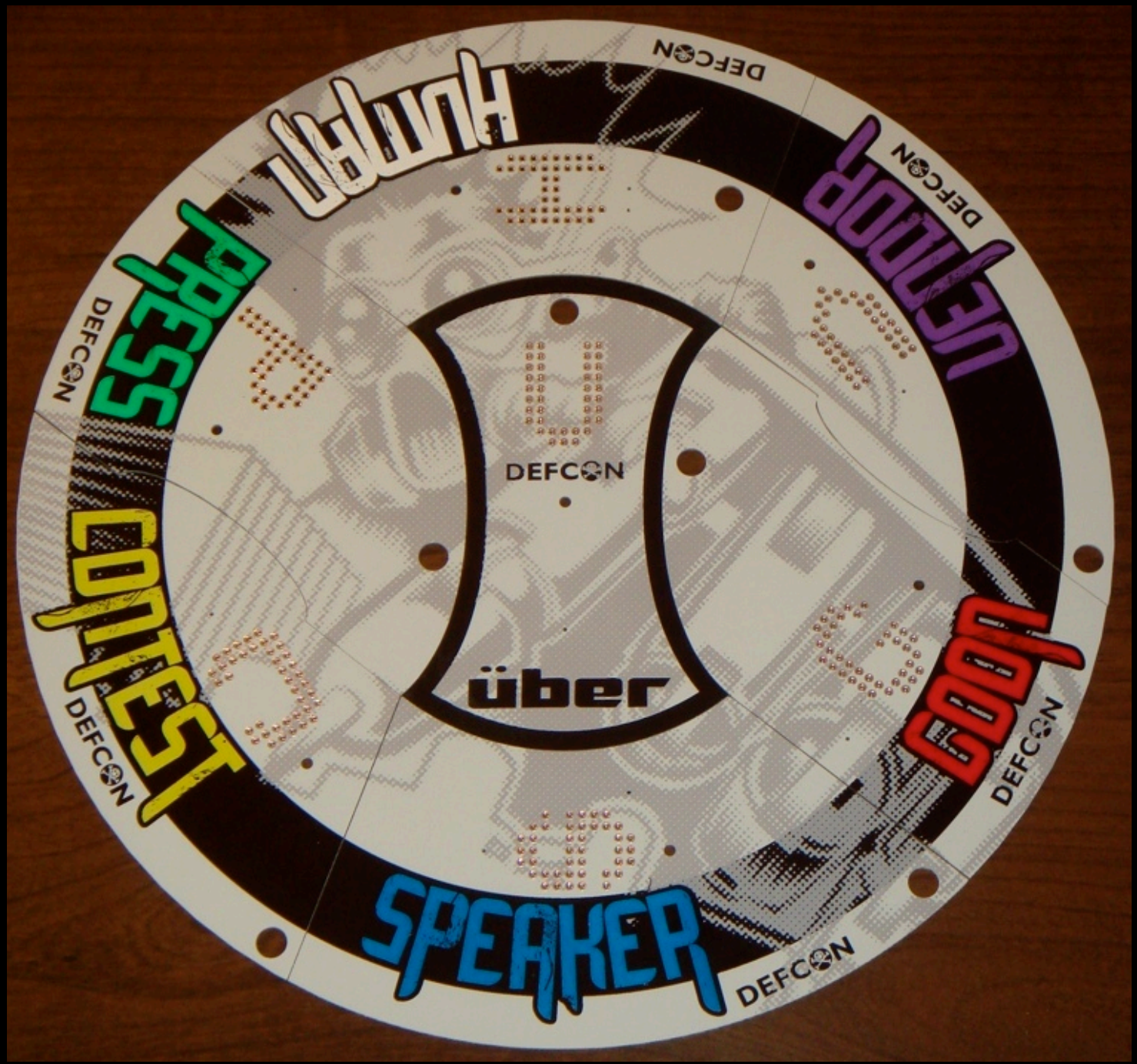
Retrospective: DEFCON 16



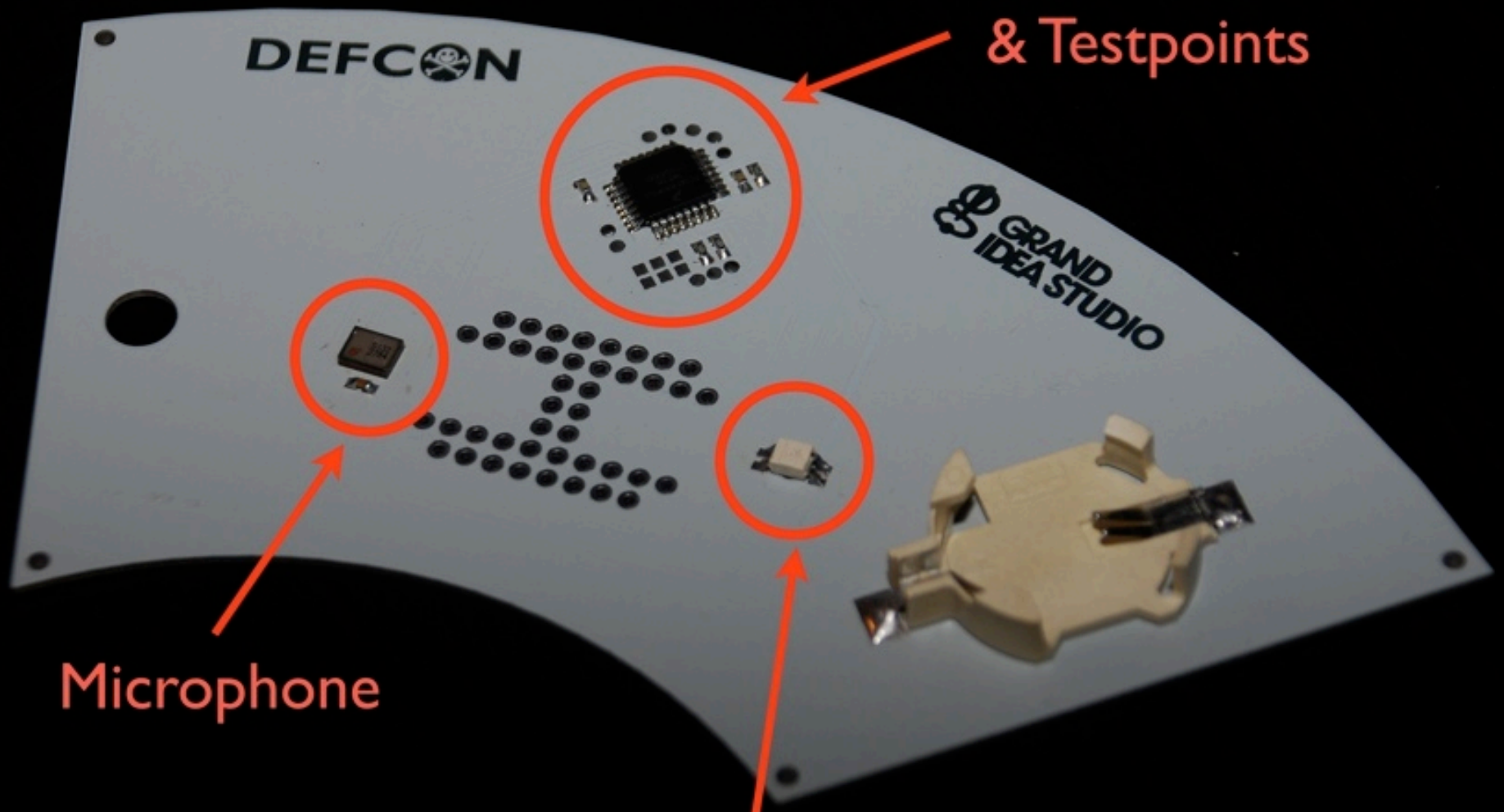
Retrospective: DEFCON 16



Retrospective: DEFCON 17



Retrospective: DEFCON 17



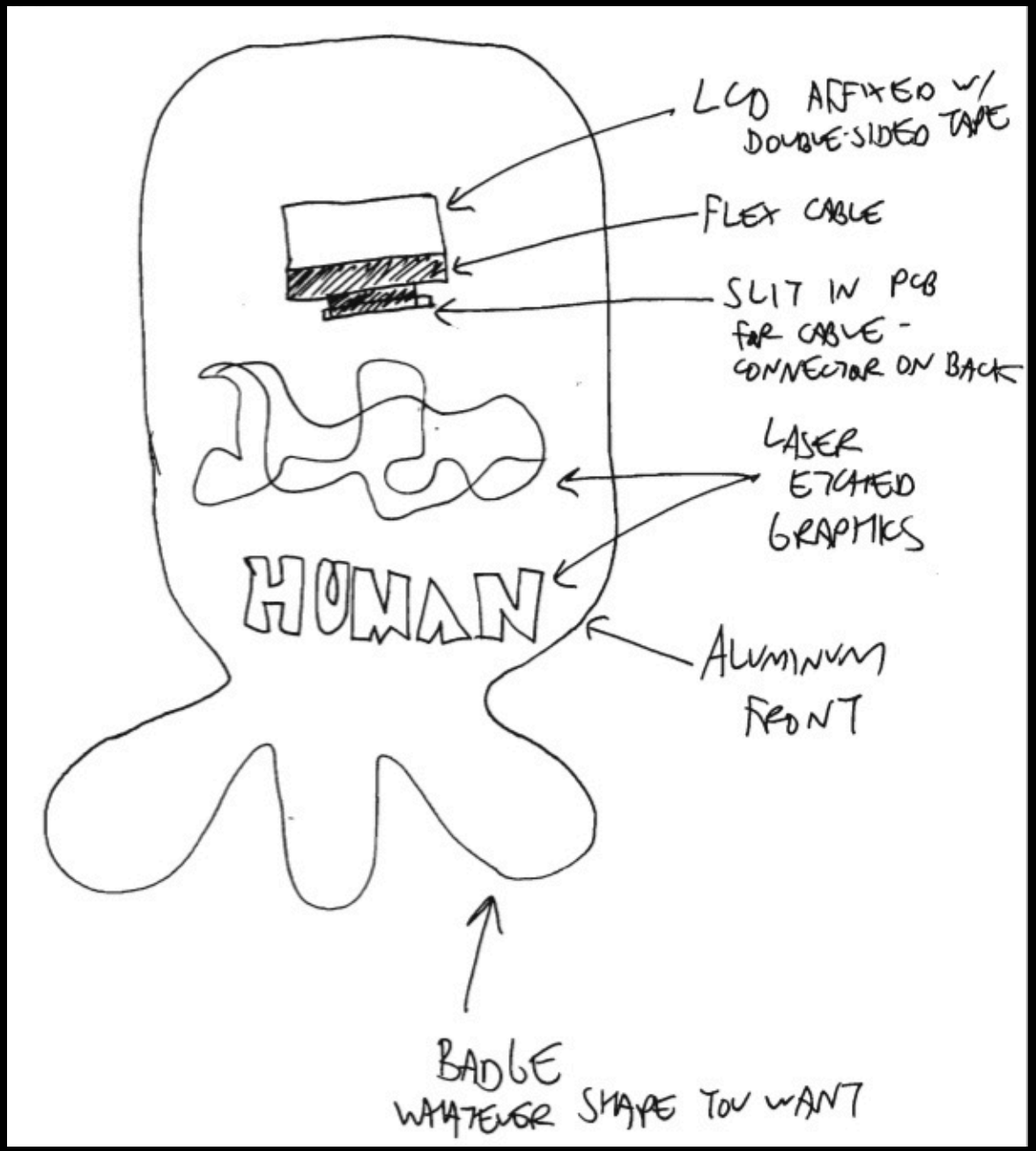
Badges by Christmas?

Timeline

- ★ Fall 2009: Initial brainstorming (DT, Black Beetle, Neil)
- ★ January 2010: Preliminary design & parts selection
- ★ January: Prototype hardware design
- ★ February: Low-level firmware completed
- ★ February: Production design finalized
- ★ March: Production component orders
- ★ April-May: Finish firmware
- ★ June: Program microcontrollers (Avnet)
- ★ June-July: Badge fabrication, assembly & test (e-Teknet)
- ★ July: Badges shipped to DEFCON (on time!)

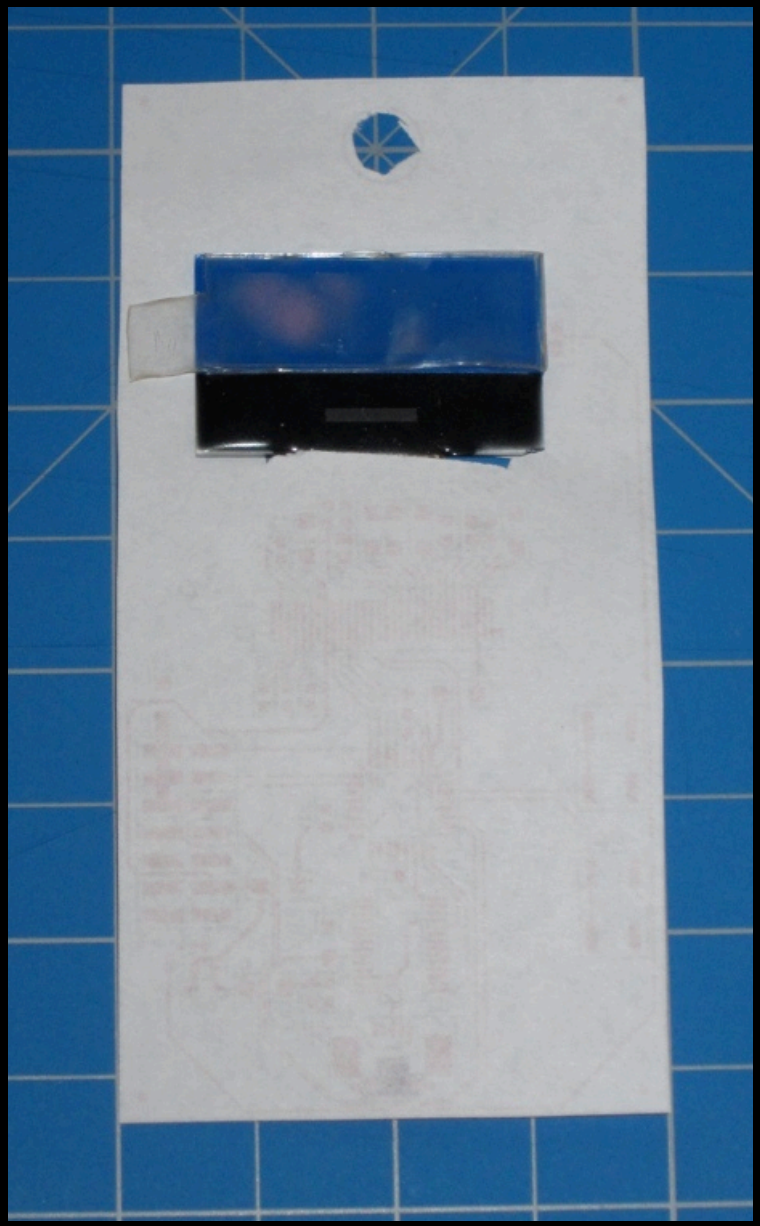
The Development Process Picture Show

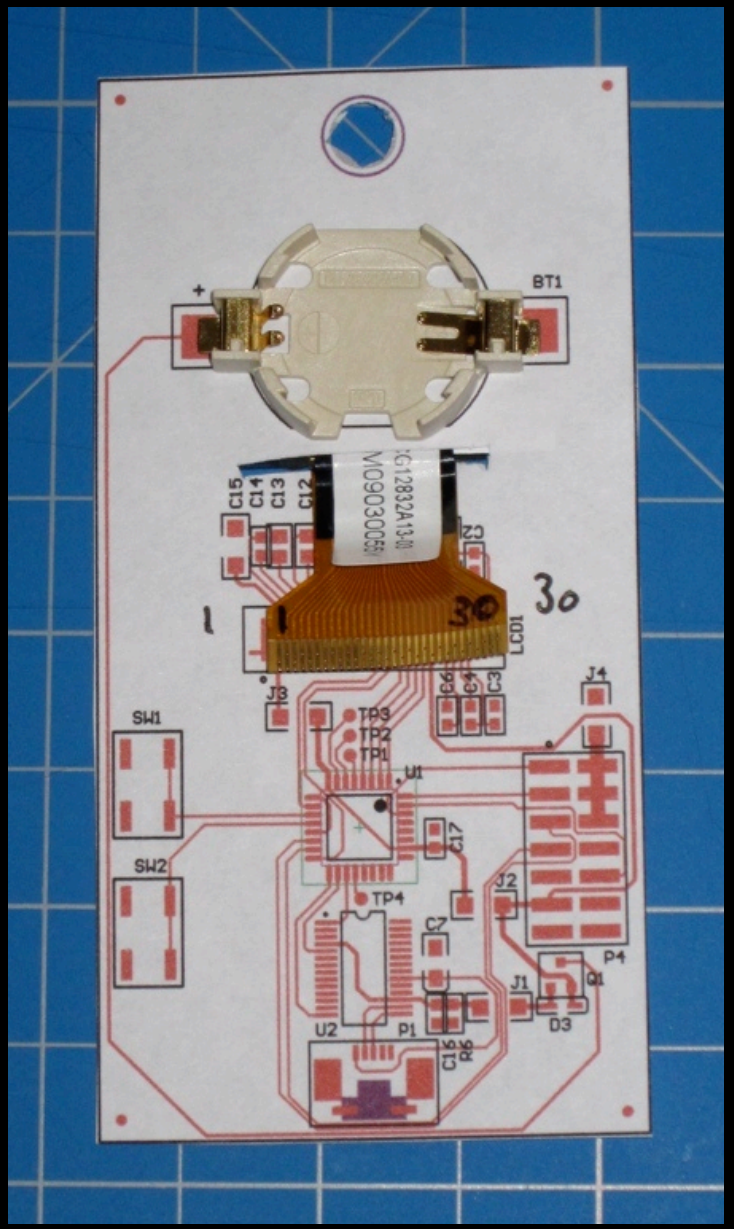
Original sketch



The Development Process
Picture Show 2

Paper mock-up



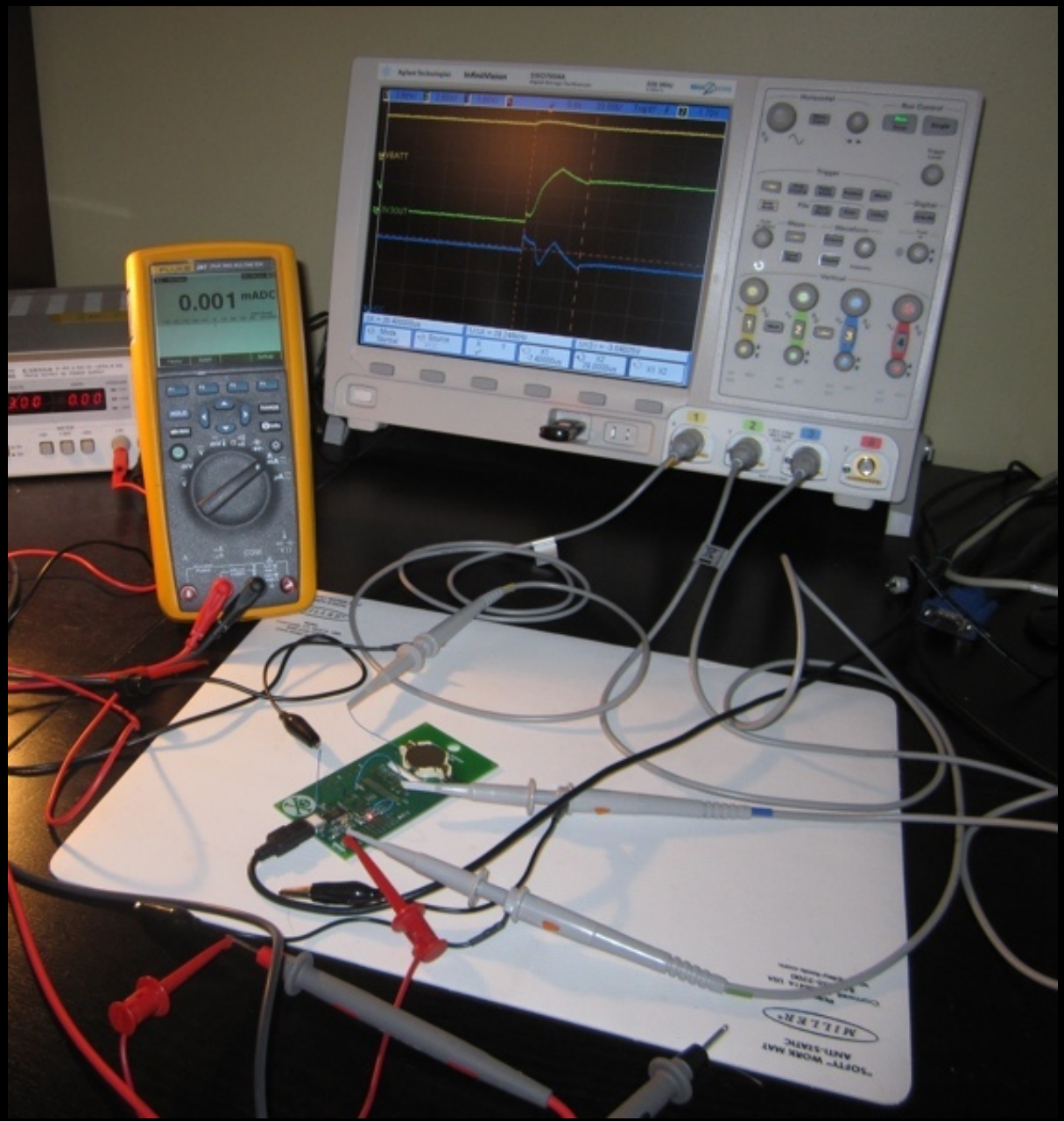


The Development Process
Picture Show 3

Prototype hardware

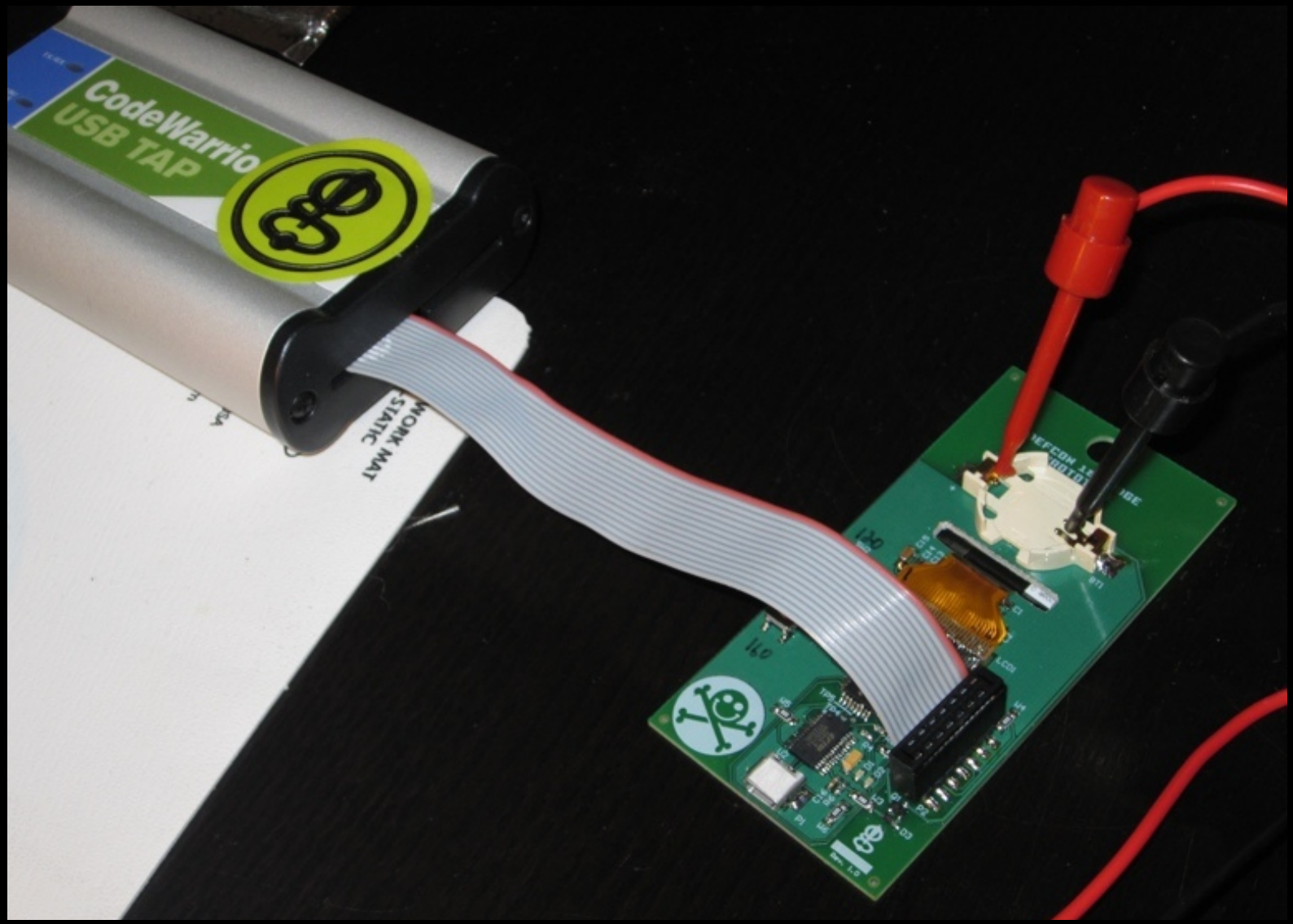
The Development Process
Picture Show 4

Testing hardware



The Development Process
Picture Show 5

Writing low-level drivers

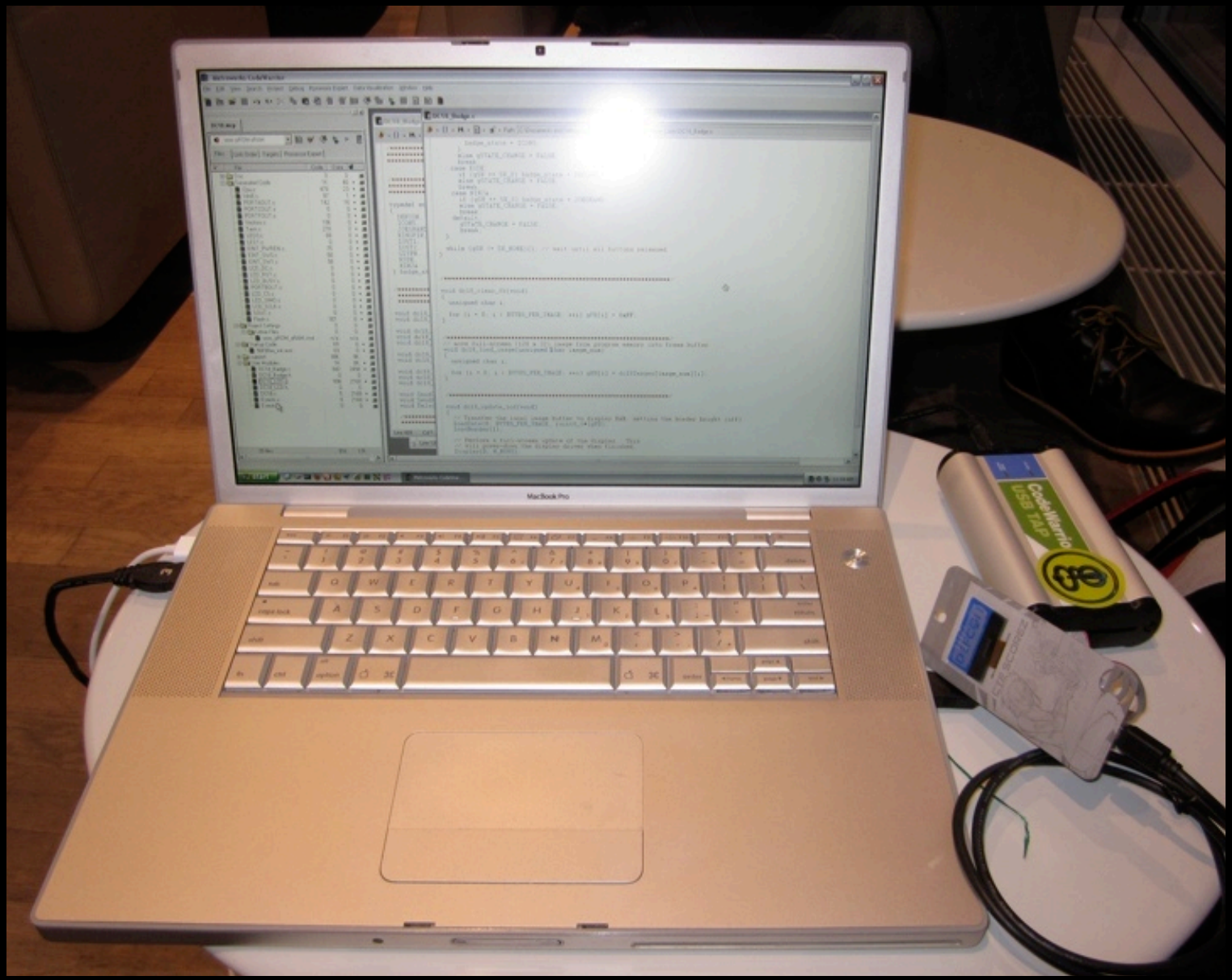


The Development Process
Picture Show 6

It works!

The Development Process Picture Show 7

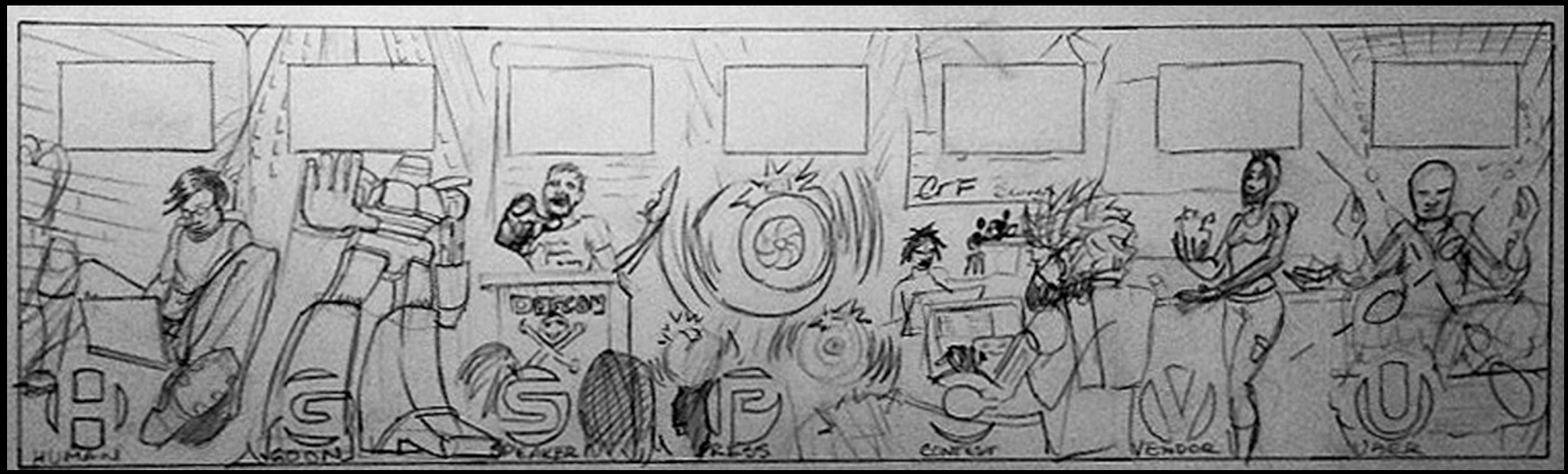
Final firmware development



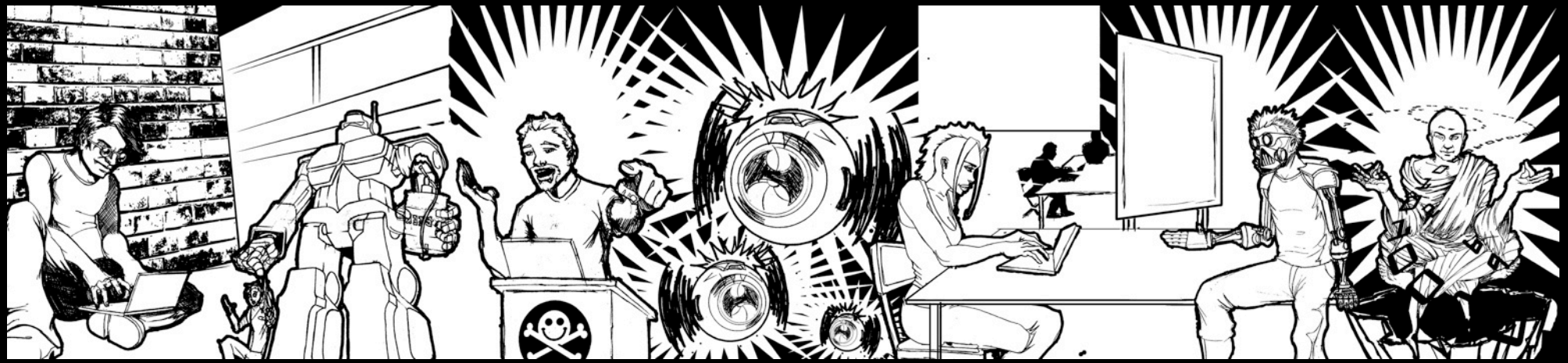
Laser Engraving!

- ★ 0.040" single-sided aluminum substrate PCB
- ★ Killer graphics by Neil, the DEFCON resident artist
- ★ Difficult to find a vendor that would take on this work
  - ★ ...and do it for an affordable price
  - ★ On aluminum? Yes. On circuit Board? No.

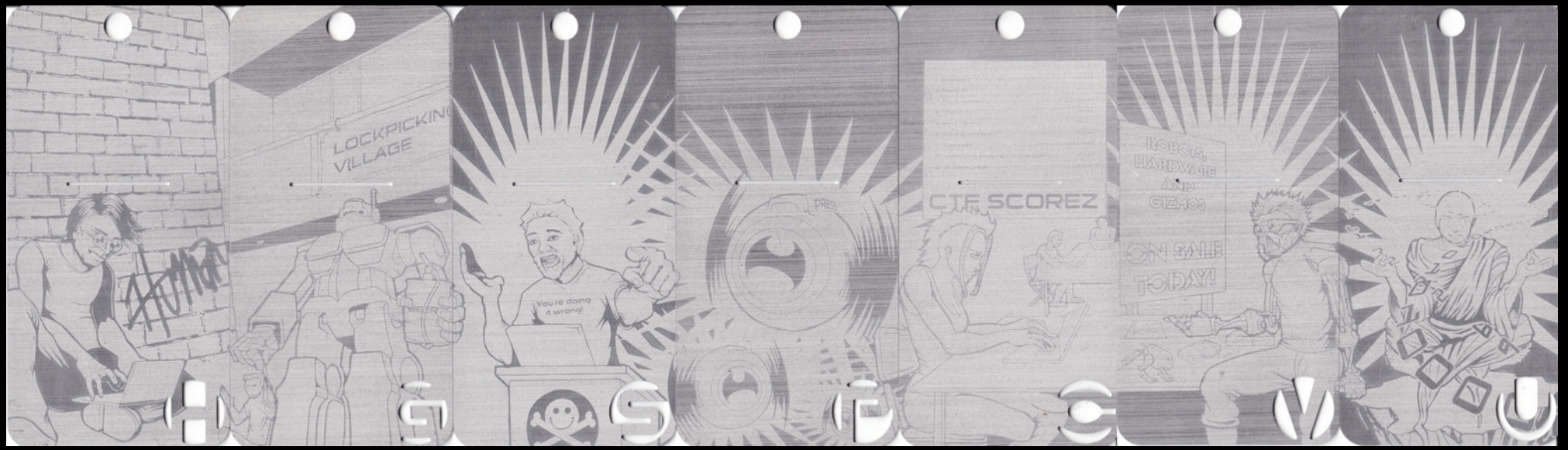
Laser Engraving! 2



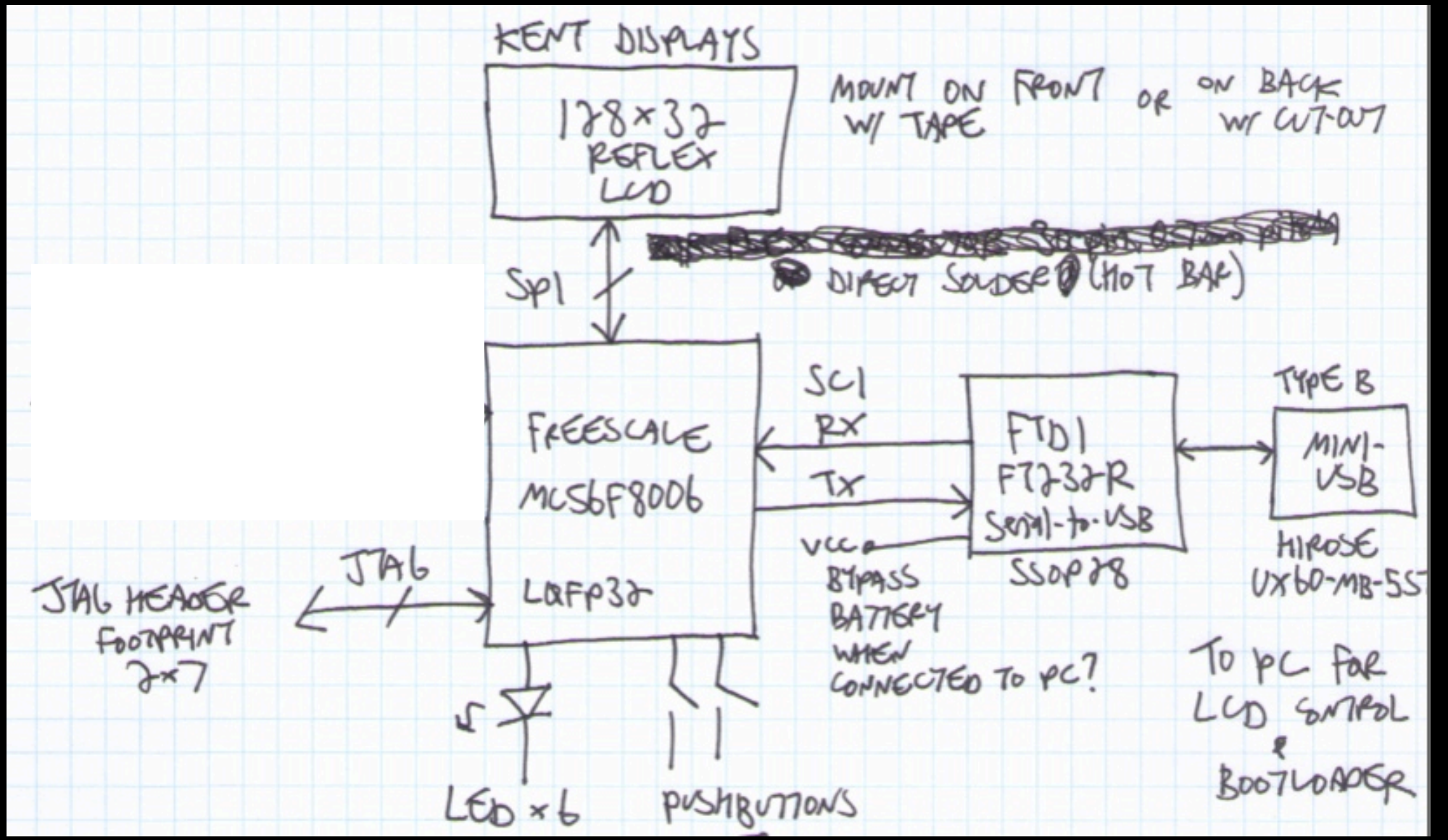
Laser Engraving! 3



Laser Engraving! 4



Early Block Diagram

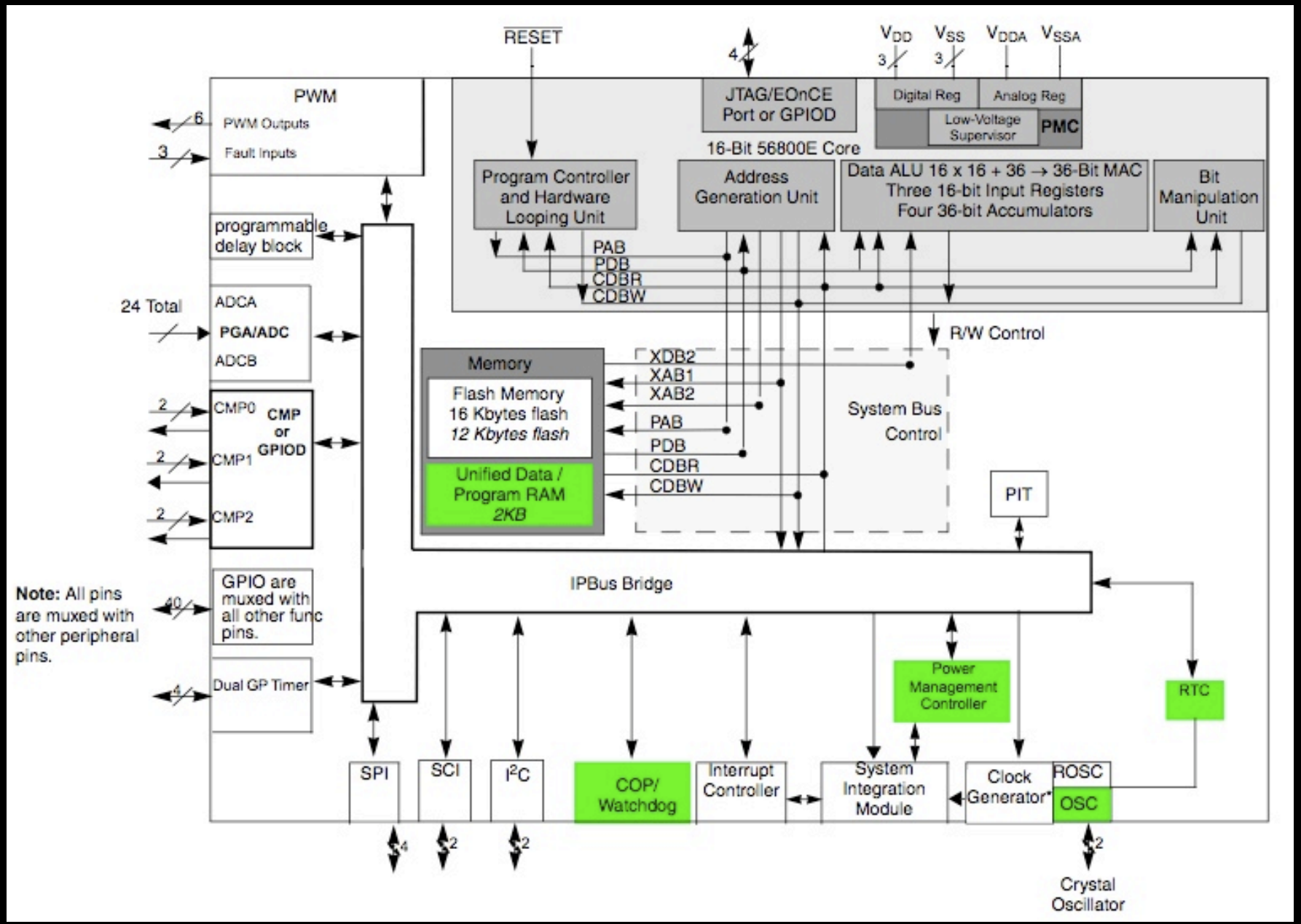


Freescale MC56F8006

- ★ Digital Signal Controller
  - ◉ Part of the 56800/E family
  - ◉ Main product page: `http://tinyurl.com/mc56f8006-info/`
  - ◉ Direct link to data sheet: `www.freescale.com/files/dsp/doc/data_sheet/MC56F8006.pdf`

- 16KB of Flash
- 2KB of RAM
- 6-channel PWM
- 18-channel, 12-bit A/D
- Timer/RTC
- 2 PGA, 3 analog comparators
- Serial communication/UART/I2C/SPI
- Up to 22 GPIO
- 32-pin LQFP, 7mm x 7mm
- 1.8V-3.6V operation

Freescale MC56F8006



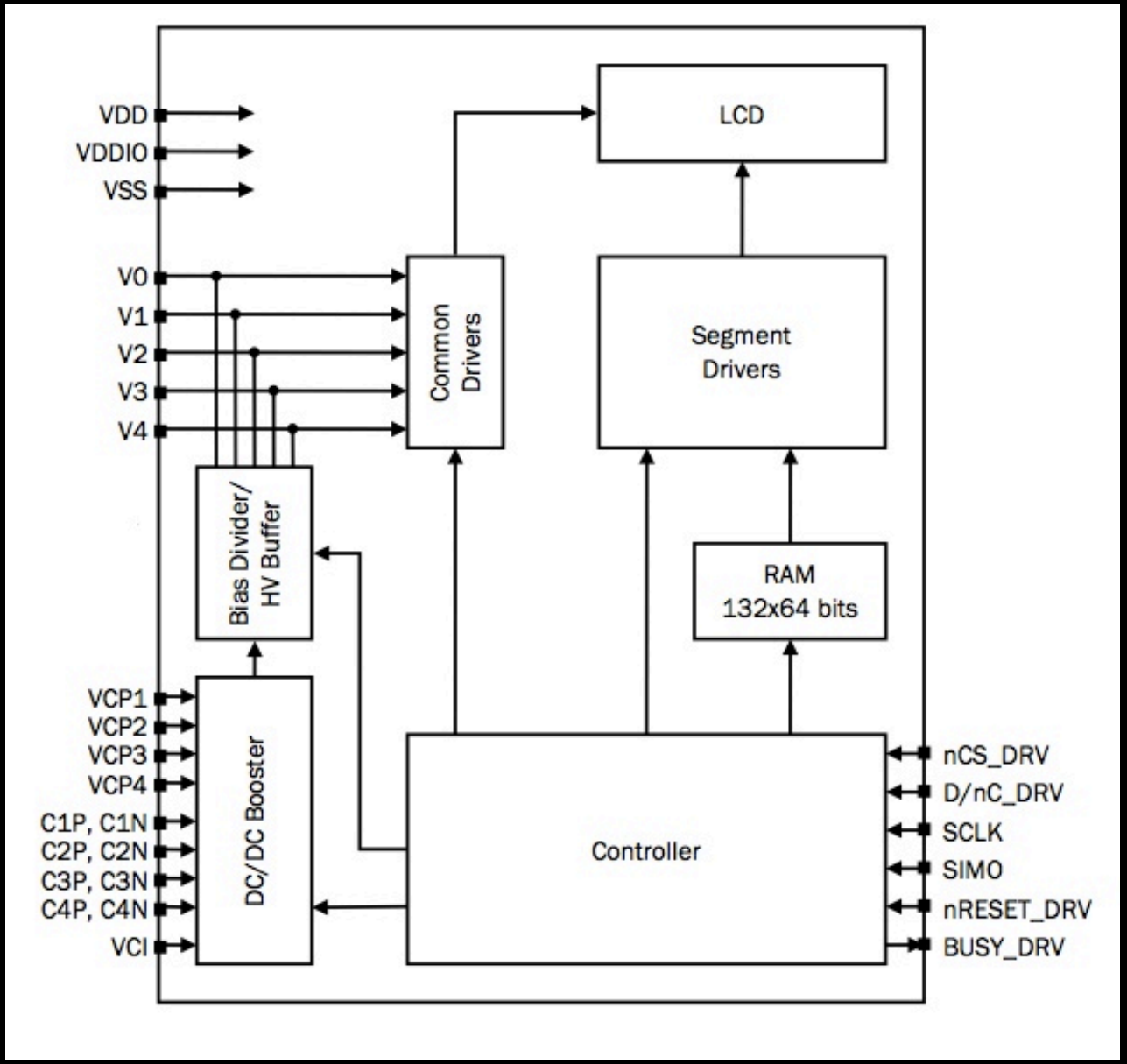
Kent Displays

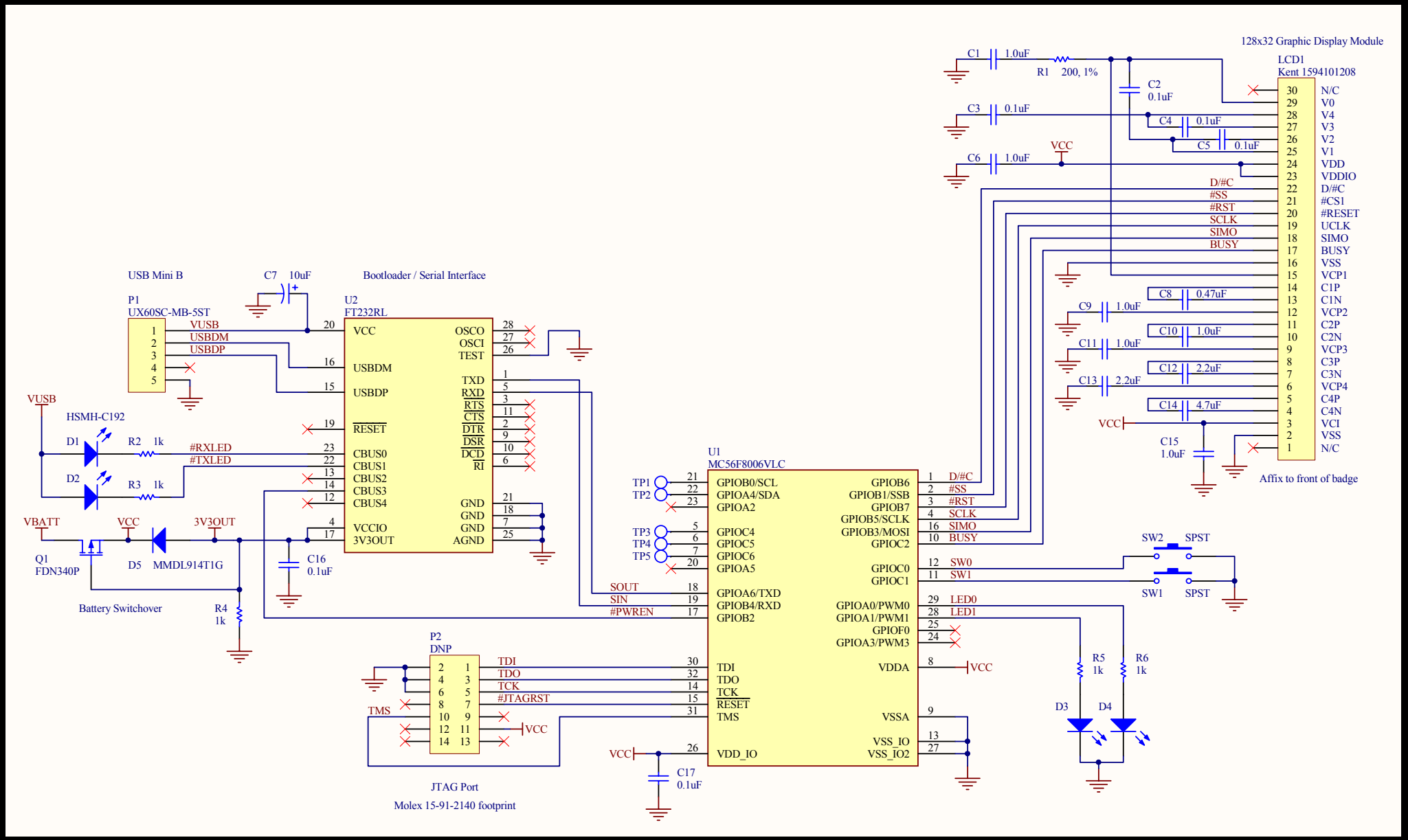
Reflex Graphic Display Module

- 128 x 32 pixels, 118 DPI
- Reflective Cholesteric LCD
- Bistable = no power or refresh needed to retain image on screen
- Control via SPI-like slave serial interface
- Full screen update ~1.7 seconds
- Affixed to badge with 3M 468MP adhesive tape
- Originally designed for use in Verbatim InSight USB Portable Hard Drive
- Not used in the first moon landing

Kent Displays

Reflex Graphic Display Module




128x32 Graphic Display Module
LCD1
Kent 1594101208
Affix to front of badge
USB Mini B
P1
UX60SC-MB-5ST
Bootloader / Serial Interface
U2
FT232RL
HSMH-C192
Battery Switchover
U1
MC56F8006VLC
JTAG Port
Molex 15-91-2140 footprint
P2
DNP
CR2032, 3V @ 225mAh
BT1
BU2032SM-JJ
VBATT

Schematic

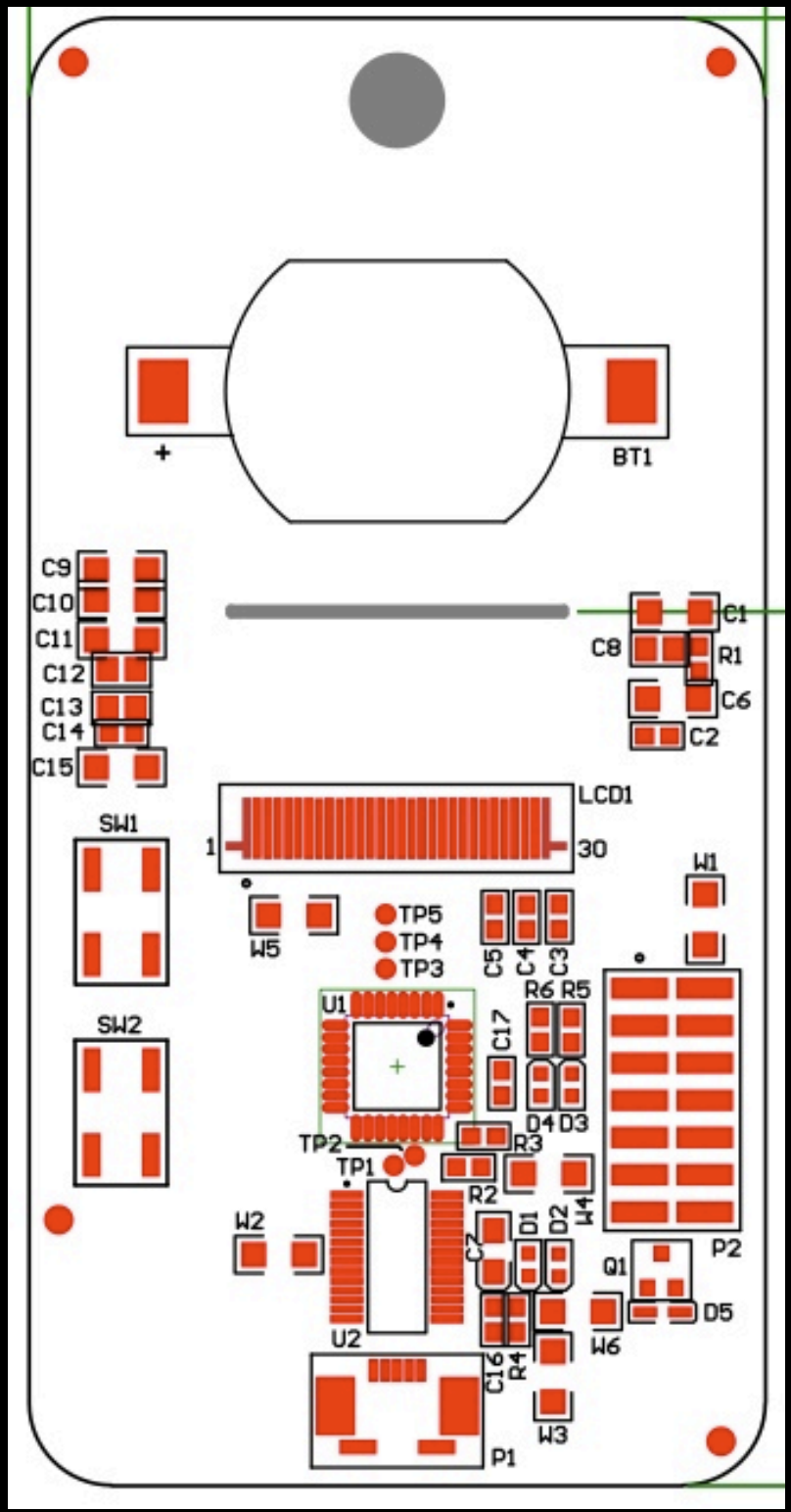
Bill-of-Materials

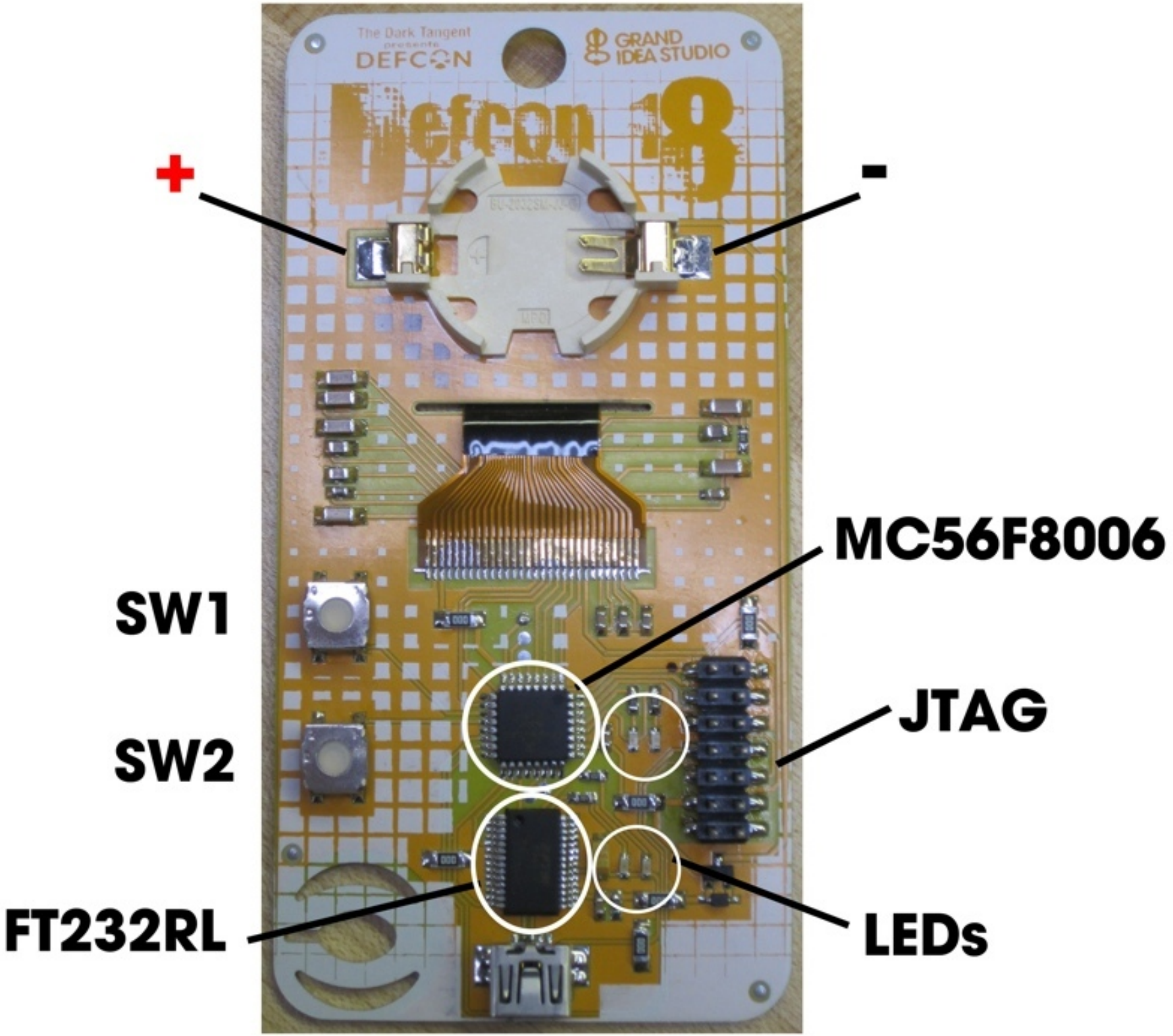
Quantity	Reference	Manufacturer	Manuf. Part #	Distributor	Distrib. Part #	Description	Unit Price	Per Badge
1	BT1	MPD	BU2032SM-JJ-GTR	Digi-Key	BU2032SM-JJ-GTR-ND	Single-cell battery holder for CR2032, SMD	$0.40000	$0.4060
1	N/A	Panasonic	CR2032	Digi-Key	P189-ND	CR2032 Lithium 3V Coin Cell Battery (225mAh)	$0.13750	$0.1375
6	C1,C6,C9,C10,C11,C15	TDK	C3216X7R1H105K	Digi-Key	445-1423-2-ND	1.0uF ceramic capacitor, 50V, X7R, 1206	$0.03100	$0.1943
6	C2,C3,C4,C5,C16,C17	Kemet	C0603C104K4RACTU	Digi-Key	399-1096-2-ND	0.1uF ceramic capacitor, 16V, X7R, 0603	$0.00240	$0.0143
1	C7	Kemet	T491A106M016AT	Newark	57K1640	10uF capacitor, 20%, 16V, tantalum, size A	$0.06900	$0.0693
1	C8	Taiyo Yuden	UMK212B7474KG-T	Digi-Key	587-1288-2-ND	0.47uF ceramic capacitor, 50V, X7R, 0805	$0.03900	$0.0524
2	C12,C13	Taiyo Yuden	TMK212BJ225KG-T	Digi-Key	587-1292-2-ND	2.2uF ceramic capacitor, 25V, X5R, 0805	$0.02600	$0.0582
1	C14	Kemet	C0603C475K8PACTU	Digi-Key	399-5503-2-ND	4.7uF ceramic capacitor, 10V, X5R, 0603	$0.03900	$0.0466
4	D1,D2,D3,D4	Avago	HSMH-C192	N/A	N/A	LED, Red, 0603, 1.8Vf, 17mcd @ 20mA	$0.02900	N/A
1	D5	ON Semiconductor	MMDL914T1G	Mouser	863-MMDL914T1G	Diode, Switching, 100Vr, 1Vf @ 10mA, 5uA Ir @ 75V, SOD-323	$0.02400	$0.0251
1	LCD1	Kent Displays	1594101208	N/A	N/A	LCD, 128x32 Reflex Graphic Display Module	$3.49000	$3.4900
1	N/A	GM Nameplate	N/A	N/A	N/A	3M 468MP adhesive tape for LCD attachment, 1” x 1/2” strips	$0.07870	$0.0822
1	P1	Hirose	UX60SC-MB-5ST(80)	Digi-Key	H11671TR-ND	Connector, Mini-USB Type B, R/A, 5 position, SMD	$0.37500	$0.4198
1	Q1	Fairchild	FDN340P	Digi-Key	FDN340PTR-ND	Transistor, MOSFET, P-Channel, 20V, 2A, SSOT3/SOT23	$0.11100	$0.1160
1	R1	Yageo	RC0603FR-07200RL	Mouser	603-RC0603FR-07200RL	200 ohm, 1%, 1/10W, 0603	$0.00200	$0.0030
5	R2,R3,R4,R5,R6	Panasonic	ERJ-3GEYJ102V	Digi-Key	P1.0KGTR-ND	1k, 5%, 1/10W, 0603	$0.00120	$0.0063
2	SW1,SW2	C&K	KSC201JLFS	Digi-Key	401-1756-2-ND	SPST tactile momentary switch, 120gf, 6.2 x 6.2mm, J-Lead	$0.16600	$0.3469
1	U1	Freescale	MC56F8006VLC	Avnet	N/A	Microcontroller/Digital Signal Controller, LQFP32	$1.50000	N/A
1	U2	FTDI	FT232RL	Mouser	895-FT232RL	USB-to-Serial UART Converter, SSOP28W	$1.89000	$1.8900
1	N/A	N/A	N/A	Avnet	N/A	Microcontroller programming service	$0.10000	$0.1000
6	W1,W2,W3,W4,W5,W6	Panasonic	ERJ-8GEY0R00V	Digi-Key	P0.0ETR-ND	Jumper, 0 ohm resistor, 1/4W, 1206	$0.00380	$0.0284
1	PCB	e-Teknet	DC18	N/A	N/A	PCB (fabrication, laser, assembly, test)	$6.63000	$6.6300

Approx. per badge cost = $14.12 (!)

Big ticket items = LCD and laser engraving ($3.84)

Assembly Drawing




+
-
MC56F8006
SW1
JTAG
SW2
FT232RL
LEDs

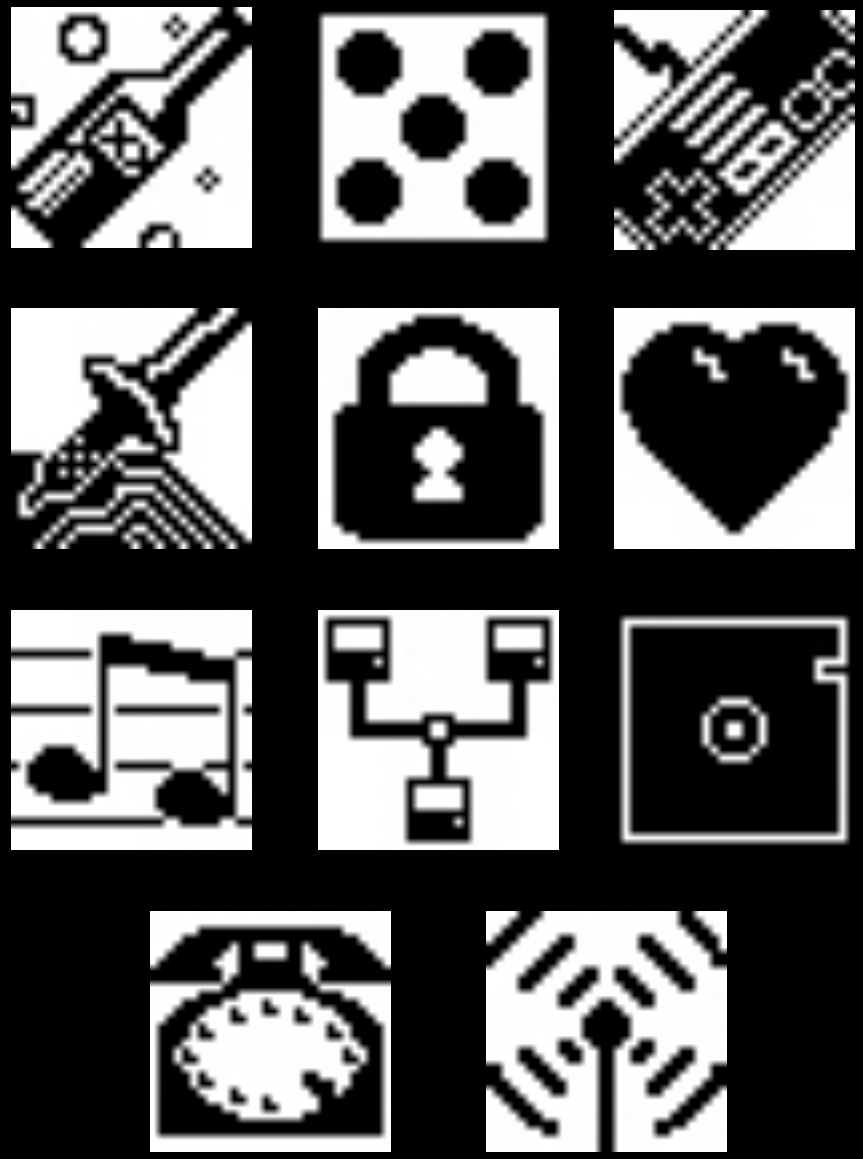
Core Badge Functionality

- ★ Glyph selection
- ★ LCD control API
- ★ Secret modes

Glyph Selection

- ★ Choose your favorite 4 glyphs/icons to show off on your LCD
- ★ Now you don't have to talk to someone to find out if they have the same interests as you :)
- ★ Enter Glyph Selection mode by pressing SW2 from the DEFCON logo
- ★ Use SW1 and SW2 to cycle through the glyphs
- ★ Press SW1 and SW2 together to select the glyph
- ★ Lather, rinse, repeat

Glyph Selection 2



LCD Control API

- ★ Control the LCD via serial commands sent over USB virtual COM port
  - 9600, 8N1
- ★ Use terminal program or scripts
- ★ Display nifty graphics and text on the LCD
- ★ No hardware hacking experience necessary!

LCD Control API 2

With USB plugged in, send '#' to enable mode
Badge will return welcome string (in ASCII)

'C' = clear frame buffer
'L aa aa vv' = load byte **vv** into frame buffer
location **aa**
ex.: **L 00 01 0A**
valid locations 0x000 to 0x1FF
see LCD data sheet pg. 9 for mem. map
0 = pixel on (dark), 1 = pixel off (light)

'U' = update LCD w/ frame buffer contents
'X' (or power cycle) = exit LCD mode

Badge will return ACK ('.') after a valid command

LCD Control API 3

PAGE	BIT	COLUMN ADDRESS 0	1	2	3	4	...	127	128	129	130	131
0	0-7	DON'T CARE					...	DON'T CARE				
1	0-6	DON'T CARE					...	DON'T CARE				
	7	B	B	B	B	B	...	B	B	B	B	
2	0-7	DON'T CARE	B	BYTE 0	BYTE 1	BYTE 2	...	BYTE 125	BYTE 126	BYTE 127	B	DON'T CARE
3	0-7	DON'T CARE	B	BYTE 128	BYTE 129	BYTE 130	...	BYTE 253	BYTE 254	BYTE 255	B	DON'T CARE
4	0-7	DON'T CARE	B	BYTE 256	BYTE 257	BYTE 258	...	BYTE 381	BYTE 382	BYTE 383	B	DON'T CARE
5	0-7	DON'T CARE	B	BYTE 384	BYTE 385	BYTE 386	...	BYTE 509	BYTE 510	BYTE 511	B	DON'T CARE
6	0		B	B	B	B	...	B	B	B	B	
	1-7	DON'T CARE					...	DON'T CARE				
7	0-7	DON'T CARE					...	DON'T CARE				

Secret Modes

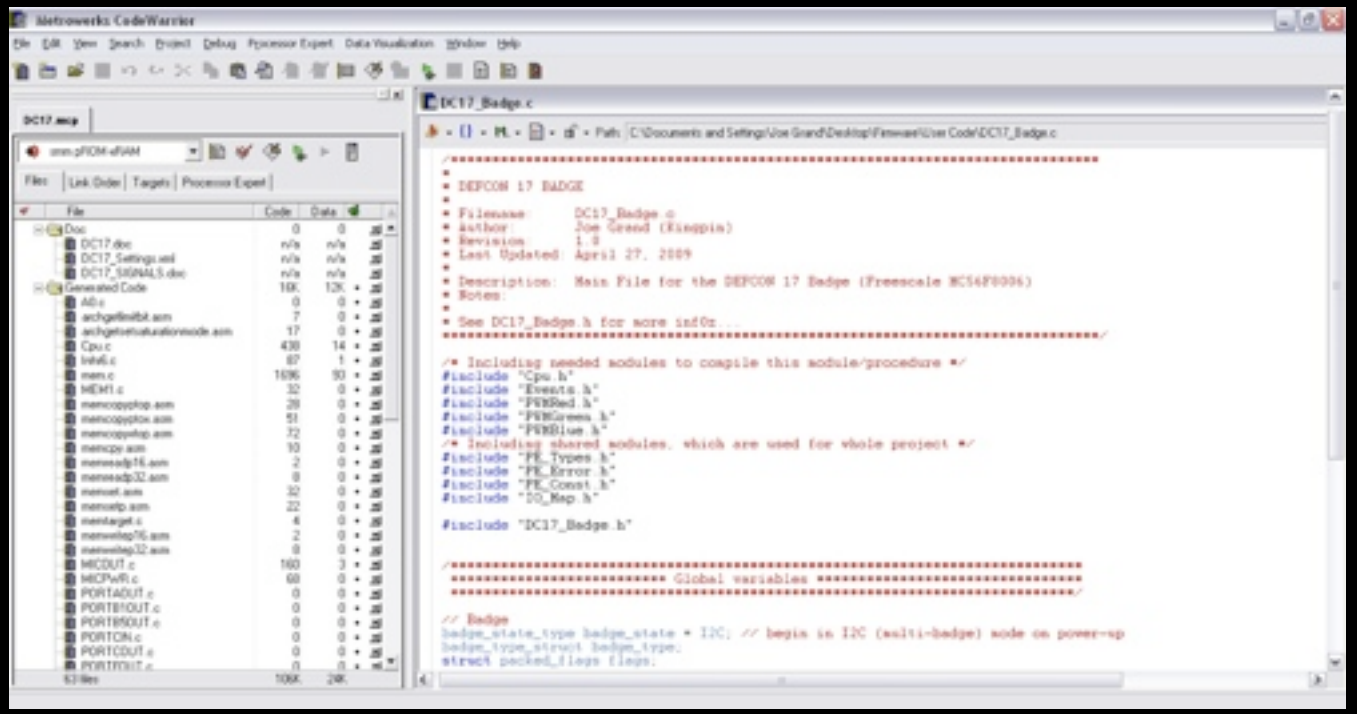
Call-for-Integration to invite DEFCON participants to hide code and/or data in the badge (March 2010)

Lots of cool stuff? Find it all!

Other Badge Stuff You Might Want To Know.

Development Environment

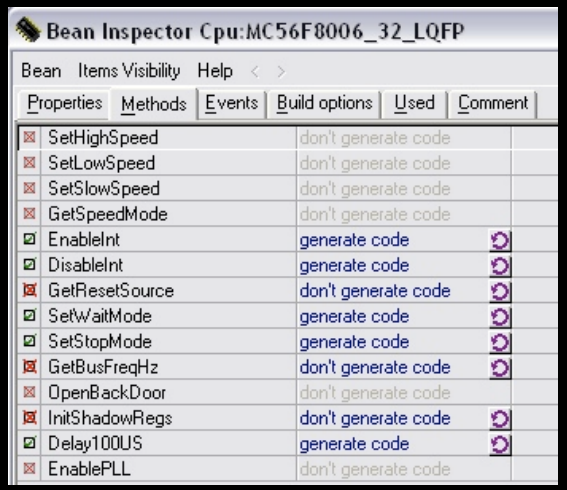
Freescale CodeWarrior for 56800/E Digital Signal Controllers

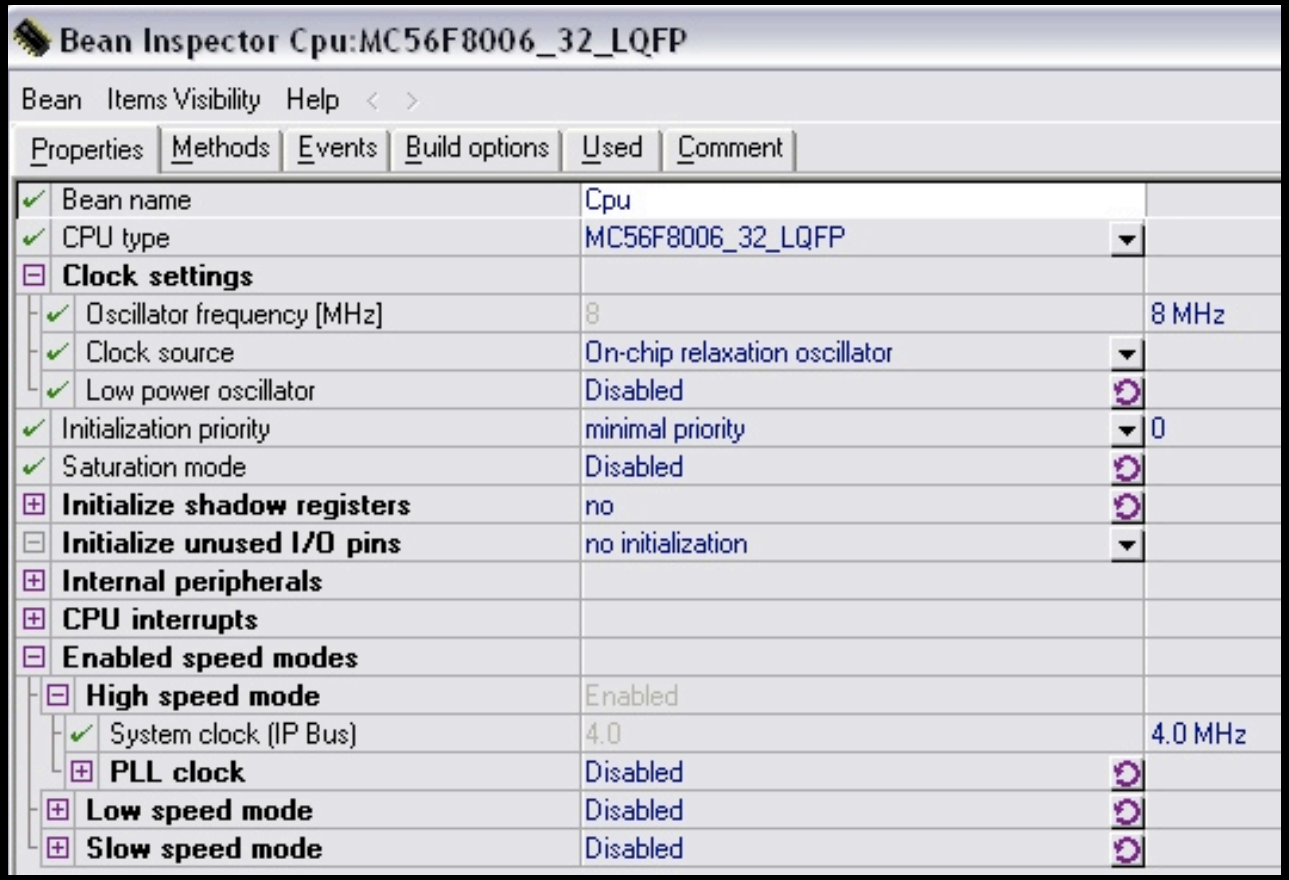


- ★ Special Edition free for up to 64KB Flash
- ★ Windows only, but works fine in a VM
- ★ All tools/materials on DEFCON CD
- ★ `http://tinyurl.com/mc56f8006-dev/`

Development Environment 2

Processor Expert





- ★ GUI for peripheral configuration
- ★ Generates required drivers/function code for desired modules

Static Serial Bootloader

- ★ USB port + terminal program = Load your own firmware onto the badge
- ★ Hold down SW1 & SW2 on power-up
  - ◉ Ideally by inserting USB cable (remove battery first)
  - ◉ Both top LEDs will remain on
  - ◉ Virtual USB COM port will appear on your machine
- ★ Use CodeWarrior dev. tools to hack/modify firmware and re-compile

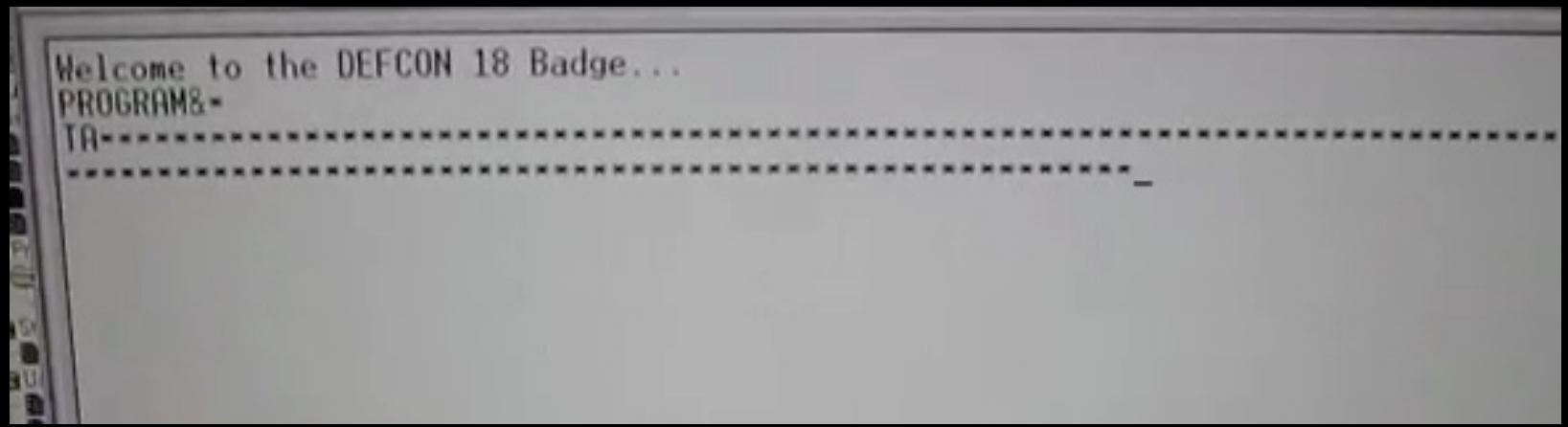
Static Serial Bootloader 2

- When modifying the User Code:
  - Need to ensure reset and COP vectors point to BOOTLOADER_ADDR (0x1B00) and not user code!
  - If you change the linker file, you'll need to re-patch it, as well
  - Read the comments in `/CODE/cpu.c` for full details

Static Serial Bootloader 3

★ Send the hex file and the badge will do the rest...

- 9600, 8N1, Xon/Xoff
- **`/output/sdm_pROM_xRAM.elf.S`**
- Typical load time ~90 seconds

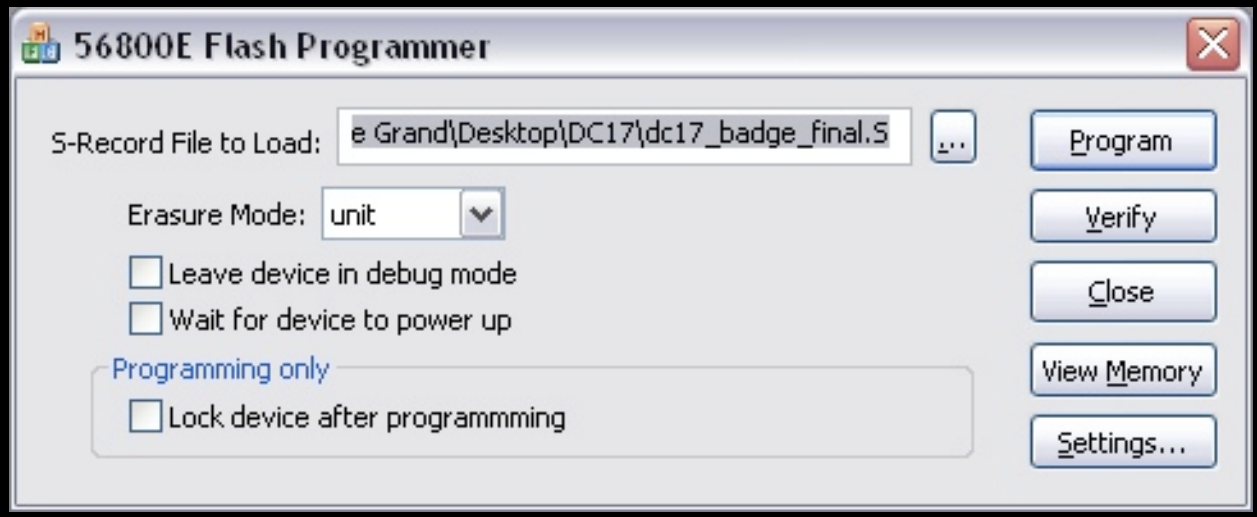


In Case Of Bricking...

- MC56F8006 JTAG interface
  - Uses CodeWarrior USB TAP hardware, `www.freescale.com/webapp/sps/site/prod_summary.jsp?code=USBTAP`
- Just like last year, but 2x7 connector footprint is provided this time
- Engineers are standing by in the Hardware Hacking Village
  - I brought ~100 connectors

In Case Of Bricking... 2

- ★ Use in conjunction with 56800E Flash Programmer tool to reload original firmware (including static bootloader)
  - `dc18-with-boot.s`
- ★ Only reprogram/debug with USB cable attached to ensure normal speed (non-reduced) operation
  - Device does not sleep when powered via USB



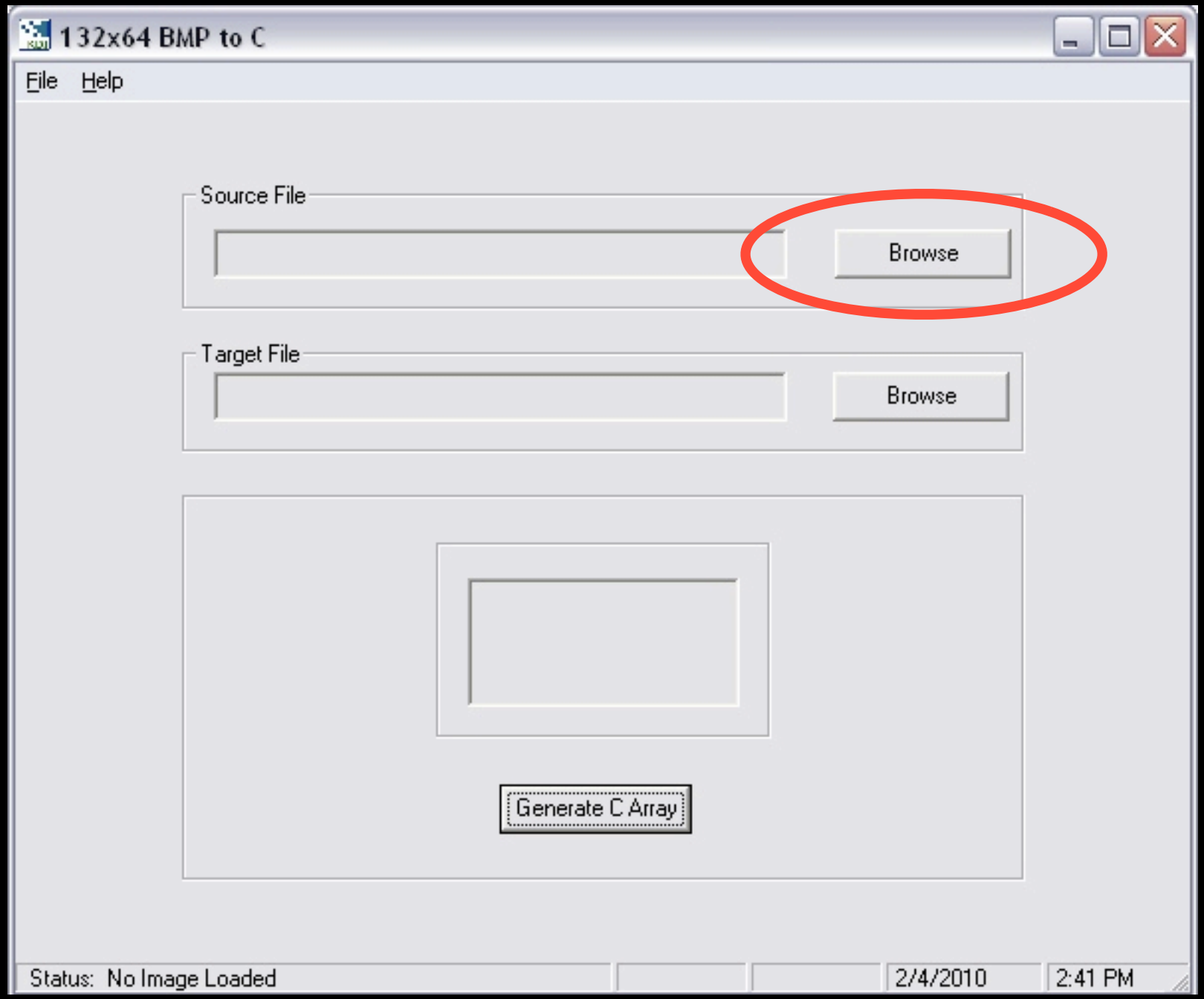
Converting BMP to C

★ Load your own graphics onto the badge
- Requires Kent Display development tools (on CD)
- Convert BMP to array and replace the one(s) in `graphics.h`

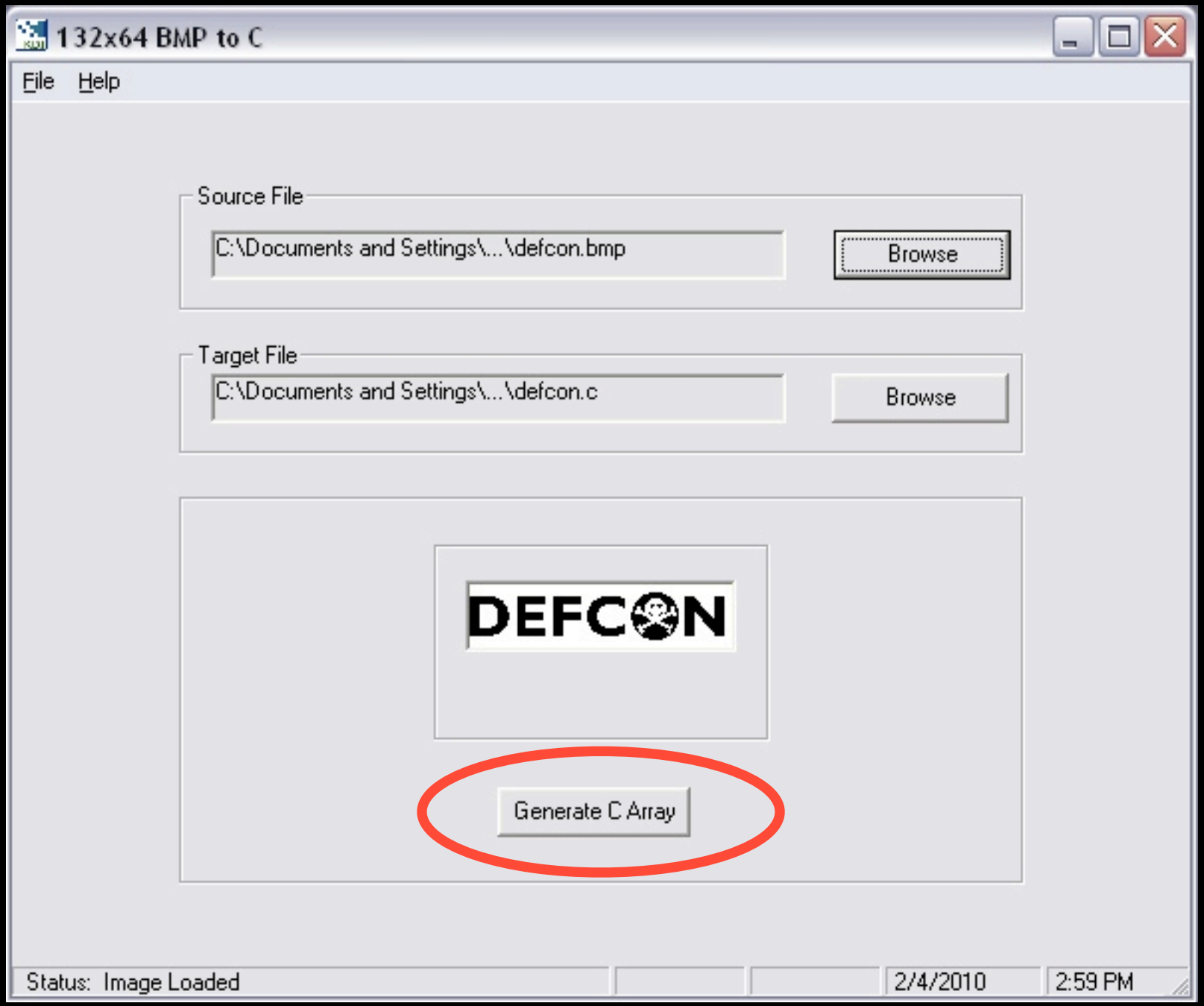
★ Maximum image size = 128 x 32 pixels

1. Convert to 132 x 32 canvas size with image at far left
2. BMP-to-C using "132x64 BMP to C" tool
3. Erase unused bytes within resultant C file
4. Copy relevant data into array

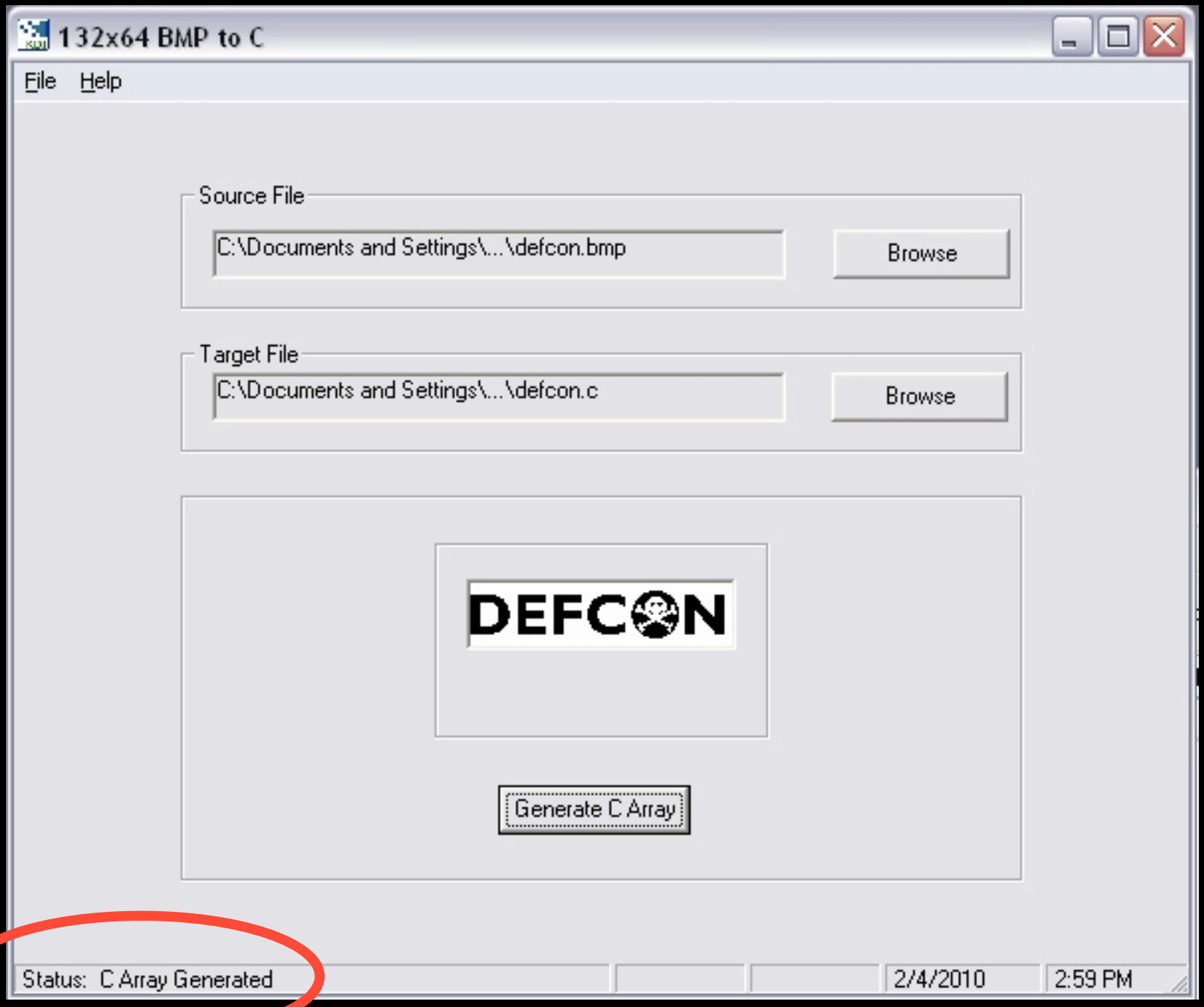
Converting BMP to C 2



Converting BMP to C 3



Converting BMP to C 4



Converting BMP to C 5

chlcd_tech.c:1

```
// Note:  uint8 must be typedef'd to an 8-bit unsigned integer in defines.h.
#include "defines.h"

// ****************************************
// chlcd_tech
// ****************************************
const uint8 chlcd_tech[512] = {
    0xFF, 0xFF, 0xFF, 0xFF, 0x07, 0x03, 0xF9, 0xFD, 0xFD, 0xF9, 0xFB, 0xFF,
    0x01, 0x01, 0xCF, 0xE7, 0xE7, 0x0F, 0x1F, 0xFF, 0x01, 0x01, 0xFF, 0xFF,
    0xFF, 0xFF, 0xFF, 0x07, 0x03, 0xF9, 0xFD, 0xFD, 0xF9, 0xFB, 0xFF, 0x01,
    0x01, 0xFD, 0xF9, 0x03, 0x07, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
    0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFD, 0xFD, 0x01, 0x01, 0xFD, 0xFD,
    0xFF, 0x0F, 0x07, 0x87, 0xB7, 0x87, 0x8F, 0xFF, 0x0F, 0x07, 0xF7, 0xF7,
    0xE7, 0xEF, 0xFF, 0x01, 0x01, 0xE7, 0xF7, 0x07, 0x0F, 0xFF, 0x07, 0x07,
    0xF7, 0xF7, 0x07, 0x0F, 0xFF, 0x0F, 0x07, 0xF7, 0xF7, 0x07, 0x0F, 0xFF,
    0xFD, 0xFD, 0x01, 0x01, 0xFF, 0xFF, 0xFF, 0x0F, 0x07, 0xF7, 0xF7, 0x07,
    0x0F, 0xFF, 0x8F, 0x07, 0x77, 0x77, 0x07, 0x0F, 0xFF, 0x87, 0x07, 0x7F,
    0x7F, 0x07, 0x07, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
    0x07, 0x07, 0x07, 0x07, 0x07, 0x06, 0x04, 0x05, 0x05, 0x04, 0x06, 0xC7,
    0xC4, 0xC4, 0xC7, 0x07, 0x07, 0x04, 0x04, 0x07, 0x04, 0x04, 0x05, 0xC5,
    0xC5, 0xC5, 0xC7, 0xC7, 0xC6, 0xC4, 0xC5, 0x05, 0x04, 0x06, 0x07, 0xE4,
    0xE4, 0x05, 0x04, 0xE6, 0xE7, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07,
    0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x04, 0xE4, 0xE7, 0x07,
    0x07, 0x06, 0x04, 0x05, 0x05, 0x05, 0x07, 0x07, 0x06, 0x04, 0xE5, 0xE5,
    0x64, 0xE6, 0x87, 0x04, 0x04, 0x07, 0x07, 0x64, 0x64, 0x07, 0x04, 0x04,
    0x07, 0x07, 0x07, 0x04, 0x04, 0x07, 0x06, 0x04, 0x05, 0x05, 0x04, 0x06, 0x07,
    0x05, 0x65, 0x64, 0xE4, 0xE5, 0x05, 0x07, 0x06, 0x04, 0x05, 0x05, 0x04,
    0x06, 0x07, 0x07, 0x05, 0x05, 0x05, 0x04, 0x06, 0x07, 0x07, 0x05, 0x05,
    0x05, 0x04, 0x06, 0x07, 0x07, 0x07, 0x07, 0x07,
    0x00, 0x00, 0x00, 0x3C, 0x3C, 0x3C, 0x3C, 0xC0, 0xC0, 0xC0, 0xC0, 0x03,
    0x03, 0x03, 0x03, 0x3C, 0x3C, 0x3C, 0x3C, 0xFC, 0xFC, 0xFC, 0xFC, 0xFF,
    0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0xFF,
    0xFF, 0x60, 0xFE, 0x9F, 0x01, 0x00, 0xF8, 0xFE, 0x86, 0x06, 0xFE, 0xF8,
    0x00, 0xFE, 0xFE, 0x86, 0x06, 0x06, 0xFE, 0xF8, 0x00, 0x06, 0xFF, 0xFF, 0x06,
    0x06, 0x06, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
    0x00, 0x01, 0xFF, 0xFE, 0x00, 0x06, 0x06, 0xFE, 0xFE, 0x00, 0x00, 0x00,
    0x78, 0xFE, 0x86, 0x06, 0x06, 0x06, 0x00, 0xFE, 0xFE, 0x06, 0x06, 0xFE,
    0xF8, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x86, 0x86, 0x86,
    0xFE, 0xF8, 0x00, 0xFE, 0xFE, 0x00, 0x00, 0xFE, 0xFE, 0x00, 0x78, 0xFE,
    0x86, 0x86, 0x86, 0x06, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x03, 0x03, 0x03, 0x00,
    0x00, 0x00, 0x00, 0x3C, 0x3C, 0x3C, 0x3C, 0x03, 0x03, 0x03, 0x03, 0x3F,
    0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x00, 0x00, 0x00, 0x00, 0x7F,
    0x7F, 0x00, 0x07, 0x7F, 0x78, 0x00, 0x1F, 0x7F, 0x61, 0x61, 0x61, 0x01,
    0x00, 0x7F, 0x7F, 0x00, 0x00, 0x7F, 0x7F, 0x00, 0x00, 0x1F, 0x7F, 0x60,
    0x60, 0x60, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7F, 0x7F,
    0x60, 0x78, 0x1F, 0x07, 0x00, 0x60, 0x60, 0x7F, 0x7F, 0x60, 0x60, 0x00,
    0x60, 0x61, 0x61, 0x61, 0x7F, 0x1E, 0x00, 0x7F, 0x7F, 0x06, 0x06, 0x07,
    0x01, 0x60, 0x60, 0x7F, 0x7F, 0x60, 0x60, 0x00, 0x1E, 0x7F, 0x61, 0x61,
    0x7F, 0x7F, 0x00, 0x01, 0x67, 0x66, 0x66, 0x7F, 0x1F, 0x00, 0x60, 0x61,
    0x61, 0x61, 0x7F, 0x1E, 0x00, 0x00, 0x00, 0x00
};
```

defcon.c:1

```
// Note:  uint8 must be typedef'd to an 8-bit unsigned integer in defines.h.
#include "defines.h"

// ****************************************
// defcon
// ****************************************
const uint8 defcon[528] = {
    0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F,
    0x3F, 0x7F, 0x7F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
    0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F,
    0x3F, 0x3F, 0xFF, 0xFF, 0xFF, 0xFF, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F,
    0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
    0xFF, 0xFF, 0xFF, 0xFF, 0x7F, 0x7F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F,
    0x3F, 0x3F, 0x3F, 0x3F, 0x7F, 0x7F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
    0xFF, 0x7F, 0x3F, 0x1F, 0x1F, 0x1F, 0x0F, 0x8F, 0x8F, 0x8F, 0x8F, 0x8F,
    0x0F, 0x1F, 0x1F, 0x3F, 0x3F, 0x7F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
    0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x7F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
    0xFF, 0xFF, 0xFF, 0x3F, 0x3F, 0x3F, 0x3F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC,
    0xF8, 0xF0, 0x00, 0x00, 0x00, 0x00, 0x03, 0x07, 0x3F, 0xFF, 0xFF, 0xFF,
    0x00, 0x00, 0x00, 0x00, 0x3C, 0x3C, 0x3C, 0x3C, 0x3C, 0x3C, 0x3C, 0x3C,
    0x3C, 0xFC, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3C,
    0x3C, 0x3C, 0x3C, 0x3C, 0x3C, 0x3C, 0x3C, 0x3C, 0xFF, 0xFF, 0xFF, 0x1F,
    0x07, 0x01, 0x00, 0x00, 0xC0, 0xF0, 0xF8, 0xF8, 0xFC, 0xFC, 0xFC, 0xFC,
    0xFC, 0xFC, 0xFC, 0xF8, 0xF8, 0xF0, 0xFF, 0xFF, 0xFF, 0x3F, 0x07, 0x61,
    0x78, 0x78, 0x60, 0xC0, 0xFC, 0xFE, 0x8F, 0x73, 0xFB, 0xFF, 0xFB, 0x73,
    0xBB, 0xFE, 0xBC, 0xC0, 0xC0, 0x78, 0xF8, 0xC3, 0x0F, 0xFF, 0xFF, 0xFF,
    0x00, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x80, 0x01, [illegible]
    0x7F, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x01, 0xFF, 0xFF, 0xFF, 0xFF,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x3F, 0x3F, 0x3F, [illegible]
    0x1F, 0x0F, 0x00, 0x00, 0x80, 0xC0, 0xE0, 0xFC, 0xFF, 0xFF, 0xFF, 0xFF,
    0x00, 0x00, 0x00, 0x00, 0x3C, 0x3C, 0x3C, 0x3C, 0x3C, 0x3C, 0x3C, 0x3C,
    0x3C, 0x3F, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFC,
    0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFF, 0xFF, 0xFF, 0xF8,
    0xE0, 0x00, 0x00, 0x00, 0x03, 0x0F, 0x1F, 0x1F, 0x3F, 0x3F, 0x3F, 0x3F,
    0x3F, 0x3F, 0x1F, 0x1F, 0x1F, 0x0F, 0xFF, 0xFF, 0xFF, 0xFC, 0xE0, 0x80,
    0x20, 0xF0, 0xE0, 0xB0, 0x30, 0x19, 0x1B, 0x0F, 0x0F, 0x07, 0x0F, 0x0F,
    0x1B, 0x19, 0x31, 0x80, 0xE0, 0x30, 0x00, 0xC0, 0xF0, 0xFF, 0xFF, 0xFF,
    0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF, [illegible]
    0xC0, 0x80, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF,
    0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, [illegible]
    0xFE, 0xFE, 0xFE, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
    0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC,
    0xFC, 0xFC, 0xFF, 0xFF, 0xFF, 0xFF, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFF,
    0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
    0xFF, 0xFF, 0xFF, 0xFF, 0xFE, 0xFE, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC,
    0xFC, 0xFC, 0xFC, 0xFE, 0xFE, 0xFE, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
    0xFF, 0xFE, 0xFD, 0xF8, 0xF8, 0xF8, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0,
    0xF0, 0xF8, 0xF8, 0xFC, 0xFD, 0xFE, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
    0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFF, 0xFF, [illegible]
    0xFF, 0xFF, 0xFF, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFF, 0xFF, 0xFF, 0xFF
};
```

Converting BMP to C 6

chlcd_tech.c:1

```
// Note:  uint8 must be typedef'd to an 8-bit unsigned integer in defines.h.
#include "defines.h"

// *******************************************
// chlcd_tech
// *******************************************
const uint8 chlcd_tech[512] = {
    0xFF, 0xFF, 0xFF, 0xFF, 0x07, 0x03, 0xF9, 0xFD, 0xFD, 0xF9, 0xFB, 0xFF,
    0x01, 0x01, 0xCF, 0xE7, 0xE7, 0x0F, 0x1F, 0xFF, 0x01, 0x01, 0xFF, 0xFF,
    0xFF, 0xFF, 0xFF, 0x07, 0x03, 0xF9, 0xFD, 0xFD, 0xF9, 0xFB, 0xFF, 0x01,
    0x01, 0xFD, 0xF9, 0x03, 0x07, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
    0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFD, 0xFD, 0x01, 0x01, 0xFD, 0xFD,
    0xFF, 0x0F, 0x07, 0xB7, 0xB7, 0x87, 0x8F, 0xFF, 0x0F, 0x07, 0xF7, 0xF7,
    0xE7, 0xEF, 0xFF, 0x01, 0x01, 0xE7, 0xF7, 0x07, 0x0F, 0xFF, 0x07, 0x07,
    0xF7, 0xF7, 0x07, 0x0F, 0xFF, 0x0F, 0x07, 0xF7, 0xF7, 0x07, 0x0F, 0xFF,
    0xFD, 0xFD, 0x01, 0x01, 0xFF, 0xFF, 0xFF, 0x0F, 0x07, 0xF7, 0xF7, 0x07,
    0x0F, 0xFF, 0x8F, 0x07, 0x77, 0x77, 0x07, 0x0F, 0xFF, 0x87, 0x07, 0x7F,
    0x7F, 0x07, 0x07, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
    0x07, 0x07, 0x07, 0x07, 0x07, 0x06, 0x04, 0x05, 0x05, 0x04, 0x06, 0xC7,
    0xC4, 0xC4, 0xC7, 0x07, 0x07, 0x04, 0x04, 0x07, 0x04, 0x04, 0x05, 0xC5,
    0xC5, 0xC5, 0xC7, 0xC7, 0xC6, 0xC4, 0xC5, 0x05, 0x04, 0x06, 0x07, 0xE4,
    0xE4, 0x05, 0x04, 0xE6, 0xE7, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07,
    0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x04, 0xE4, 0xE7, 0x07,
    0x07, 0x06, 0x04, 0x05, 0x05, 0x05, 0x07, 0x07, 0x06, 0x04, 0xE5, 0xE5,
    0x64, 0xE6, 0x07, 0x04, 0x04, 0x07, 0x07, 0x64, 0x64, 0x07, 0x04, 0x04,
    0x07, 0x07, 0x07, 0x04, 0x04, 0x07, 0x06, 0x04, 0x05, 0x05, 0x04, 0x06, 0x07,
    0x05, 0x65, 0x64, 0xE4, 0xE5, 0x05, 0x07, 0x06, 0x04, 0x05, 0x05, 0x04,
    0x06, 0x07, 0x07, 0x05, 0x05, 0x05, 0x04, 0x06, 0x07, 0x07, 0x05, 0x05,
    0x05, 0x04, 0x06, 0x07, 0x07, 0x07, 0x07, 0x07,
    0x00, 0x00, 0x00, 0x3C, 0x3C, 0x3C, 0x3C, 0xC0, 0xC0, 0xC0, 0xC0, 0x03,
    0x03, 0x03, 0x03, 0x3C, 0x3C, 0x3C, 0x3C, 0xFC, 0xFC, 0xFC, 0xFC, 0xFF,
    0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0xFF,
    0xFF, 0x60, 0xFE, 0x9F, 0x01, 0x00, 0xF8, 0xFE, 0x86, 0x86, 0xFE, 0xF8,
    0x00, 0xFE, 0xFE, 0x06, 0x06, 0xFE, 0xF8, 0x00, 0x06, 0xFF, 0xFF, 0x06,
    0x06, 0x06, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
    0x00, 0x01, 0xFF, 0xFE, 0x00, 0x06, 0x06, 0xFE, 0xFE, 0x00, 0x00, 0x00,
    0x78, 0xFE, 0x86, 0x86, 0x86, 0x06, 0x00, 0xFE, 0xFE, 0x06, 0x06, 0xFE,
    0xF8, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x86, 0x86, 0x86,
    0xFE, 0xF8, 0x00, 0xFE, 0xFE, 0x00, 0x00, 0xFE, 0xFE, 0x00, 0x78, 0xFE,
    0x86, 0x86, 0x86, 0x86, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x03, 0x03, 0x03, 0x00,
    0x00, 0x00, 0x00, 0x3C, 0x3C, 0x3C, 0x3C, 0x03, 0x03, 0x03, 0x03, 0x3F,
    0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x00, 0x00, 0x00, 0x00, 0x7F,
    0x7F, 0x00, 0x07, 0x7F, 0x78, 0x00, 0x1F, 0x7F, 0x61, 0x61, 0x61, 0x01,
    0x00, 0x7F, 0x7F, 0x00, 0x00, 0x7F, 0x7F, 0x00, 0x00, 0x1F, 0x7F, 0x60,
    0x60, 0x60, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7F, 0x7F,
    0x60, 0x78, 0x1F, 0x07, 0x00, 0x60, 0x60, 0x7F, 0x7F, 0x60, 0x60, 0x00,
    0x60, 0x61, 0x61, 0x61, 0x7F, 0x1E, 0x00, 0x7F, 0x7F, 0x06, 0x06, 0x07,
    0x01, 0x60, 0x60, 0x7F, 0x7F, 0x60, 0x60, 0x00, 0x1E, 0x7F, 0x61, 0x61,
    0x7F, 0x7F, 0x00, 0x01, 0x67, 0x66, 0x66, 0x7F, 0x1F, 0x00, 0x60, 0x61,
    0x61, 0x61, 0x7F, 0x1E, 0x00, 0x00, 0x00, 0x00
};
```

defcon.c:51

```
// Note:  uint8 must be typedef'd to an 8-bit unsigned integer in defines.h.
#include "defines.h"

// *******************************************
// defcon
// *******************************************
const uint8 defcon[528] = {
    0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F,
    0x3F, 0x7F, 0x7F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
    0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F,
    0x3F, 0x3F, 0xFF, 0xFF, 0xFF, 0xFF, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F,
    0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0xFF, 0xFF, 0xFF, 0xFF,
    0xFF, 0xFF, 0xFF, 0xFF, 0x7F, 0x7F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F,
    0x3F, 0x3F, 0x3F, 0x3F, 0x7F, 0x7F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
    0xFF, 0x7F, 0x3F, 0x1F, 0x1F, 0x1F, 0x0F, 0x8F, 0x8F, 0x8F, 0x8F, 0x8F,
    0x0F, 0x1F, 0x1F, 0x3F, 0x3F, 0x7F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
    0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x7F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
    0xFF, 0xFF, 0xFF, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F,
    0x00, 0x00, 0x00, 0x00, 0x00, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC,
    0xF8, 0xF0, 0x00, 0x00, 0x00, 0x03, 0x07, 0x3F, 0xFF, 0xFF, 0xFF, 0xFF,
    0x00, 0x00, 0x00, 0x00, 0x3C, 0x3C, 0x3C, 0x3C, 0x3C, 0x3C, 0x3C, 0x3C,
    0x3C, 0xFC, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3C,
    0x3C, 0x3C, 0x3C, 0x3C, 0x3C, 0x3C, 0x3C, 0x3C, 0xFF, 0xFF, 0xFF, 0x1F,
    0x07, 0x01, 0x00, 0x00, 0xC0, 0xF0, 0xF8, 0xF8, 0xFC, 0xFC, 0xFC, 0xFC,
    0xFC, 0xFC, 0xFC, 0xF8, 0xF8, 0xF0, 0xFF, 0xFF, 0xFF, 0x3F, 0x07, 0x61,
    0x78, 0x78, 0x60, 0xC0, 0xFC, 0xFE, 0xBF, 0x73, 0xFB, 0xFF, 0xFB, 0x73,
    0xBB, 0xFE, 0xBC, 0xC0, 0xC0, 0x78, 0xF8, 0xC3, 0x0F, 0xFF, 0xFF, 0xFF,
    0x00, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x00, 0x01, 0x07, 0x0F, 0x1F, 0x3F,
    0x7F, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x1F,
    0x1F, 0x0F, 0x00, 0x00, 0x80, 0xC0, 0xE0, 0xFC, 0xFF, 0xFF, 0xFF, 0xFF,
    0x00, 0x00, 0x00, 0x00, 0x3C, 0x3C, 0x3C, 0x3C, 0x3C, 0x3C, 0x3C, 0x3C,
    0x3C, 0x3F, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFC,
    0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFF, 0xFF, 0xFF, 0xF8,
    0xE0, 0x80, 0x00, 0x00, 0x03, 0x0F, 0x1F, 0x1F, 0x3F, 0x3F, 0x3F, 0x3F,
    0x3F, 0x3F, 0x1F, 0x1F, 0x1F, 0x0F, 0xFF, 0xFF, 0xFF, 0xFC, 0xE0, 0x80,
    0x20, 0xF0, 0xE0, 0xB0, 0x30, 0x19, 0x1B, 0x0F, 0x0F, 0x07, 0x0F, 0x0F,
    0x1B, 0x19, 0x31, 0xB0, 0xE0, 0x30, 0x00, 0xC0, 0xF0, 0xFF, 0xFF, 0xFF,
    0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFE, 0xF8, 0xF0, 0xE0,
    0xC0, 0x80, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00,
    0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC,
    0xFE, 0xFE, 0xFE, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
    0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC,
    0xFC, 0xFC, 0xFF, 0xFF, 0xFF, 0xFF, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFF,
    0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
    0xFF, 0xFF, 0xFF, 0xFF, 0xFE, 0xFE, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC,
    0xFC, 0xFC, 0xFC, 0xFE, 0xFE, 0xFE, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
    0xFF, 0xFE, 0xFD, 0xF8, 0xF8, 0xF8, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0,
    0xF0, 0xF8, 0xF8, 0xFC, 0xFD, 0xFE, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
    0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
    0xFF, 0xFF, 0xFF, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC
};
```

Power Consumption

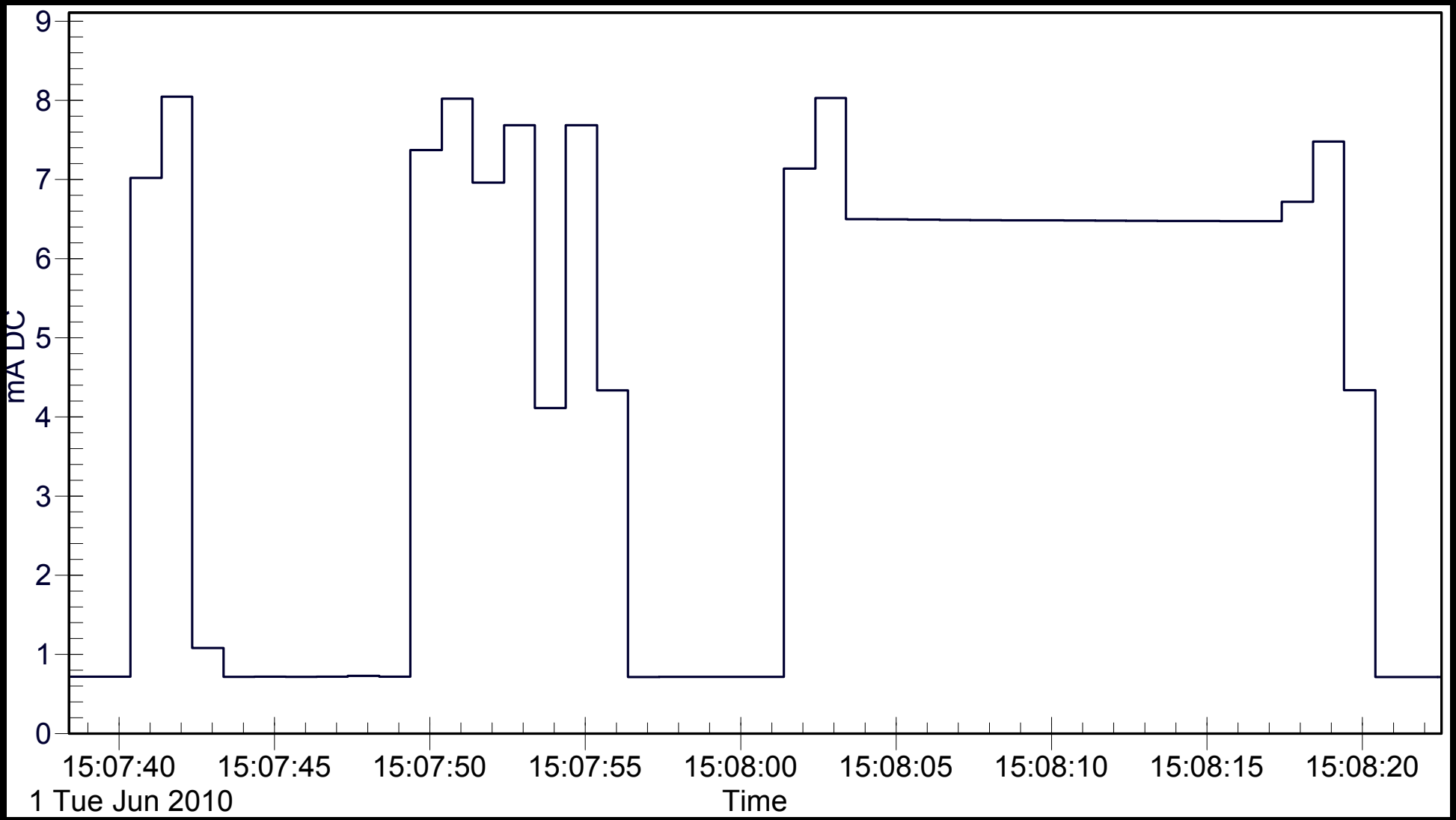
CR2032 Lithium coin cell : 3V @ 225mAh to 2V

- Idle (Wait) = 0.7mA
- Active (LCD update) = 8.3mA @ 930mS
- Serial port TX = 6.7mA

Idle mode only = 13.4 days

Typical daily use: 1H active, 23H idle = 9 days

Power Consumption 2

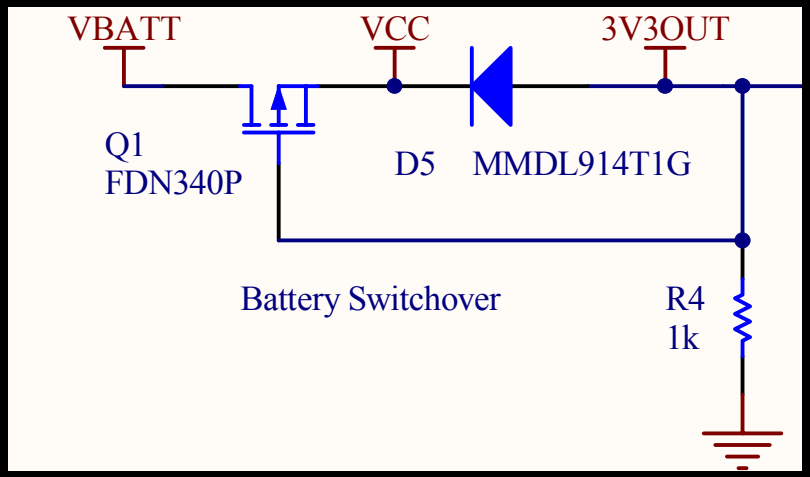


Single LCD refresh

Multiple mode changes

Serial port transmitting

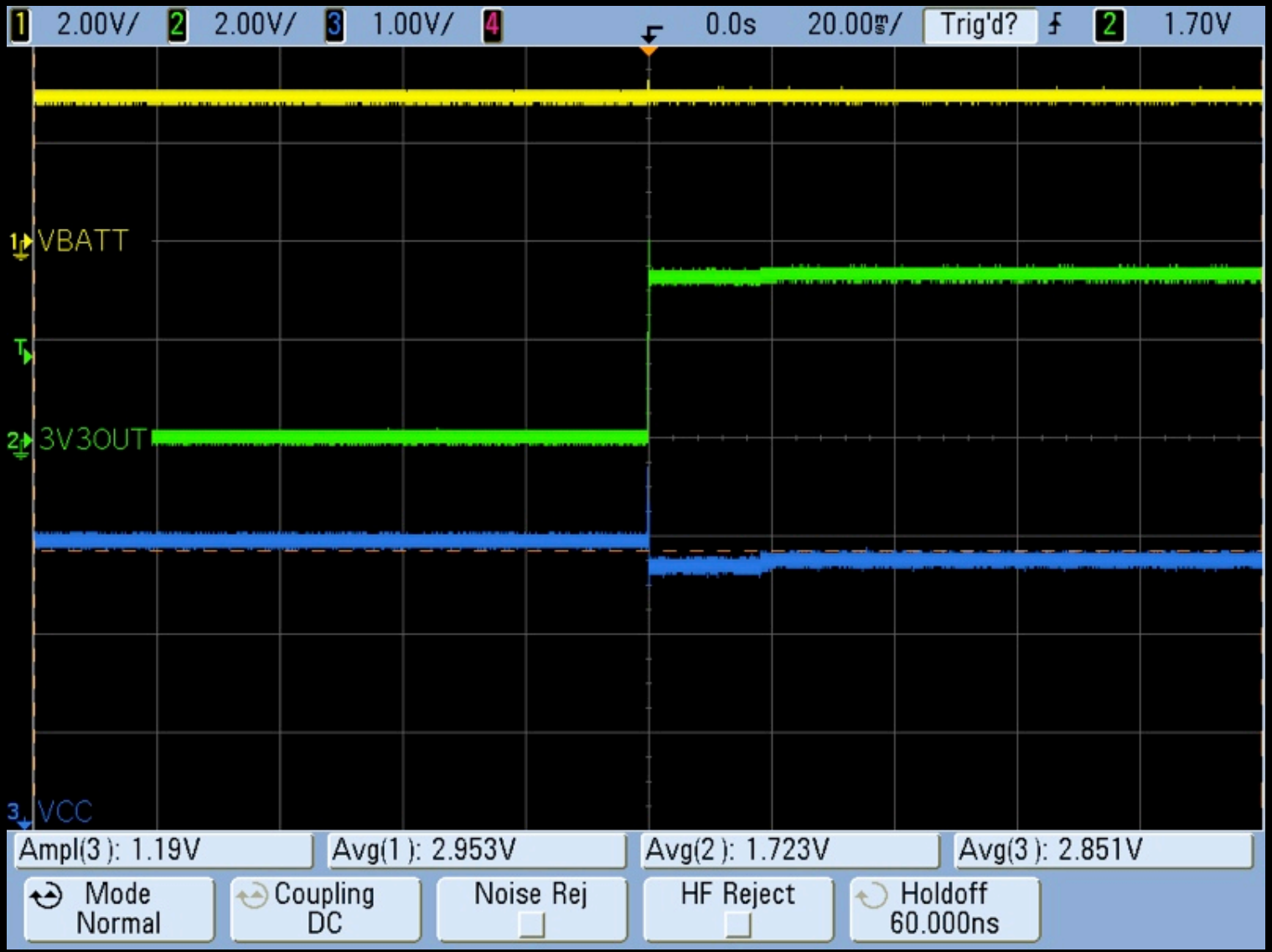
Seamless Power Switching



- P-channel MOSFET on by default via R4
- When USB plugged in, 3V3OUT (FT232) goes HIGH
- MOSFET turns off and battery is isolated from circuit
- Body diode of MOSFET prevents battery from getting reverse fed by 3V3OUT (small nA leakage is OK)
- Voltage drop across D5 causes VCC to be lower for USB-powered (2.7V) than battery powered (3V)
  - Higher forward voltage (Vf) -> lower reverse leakage (Vr)

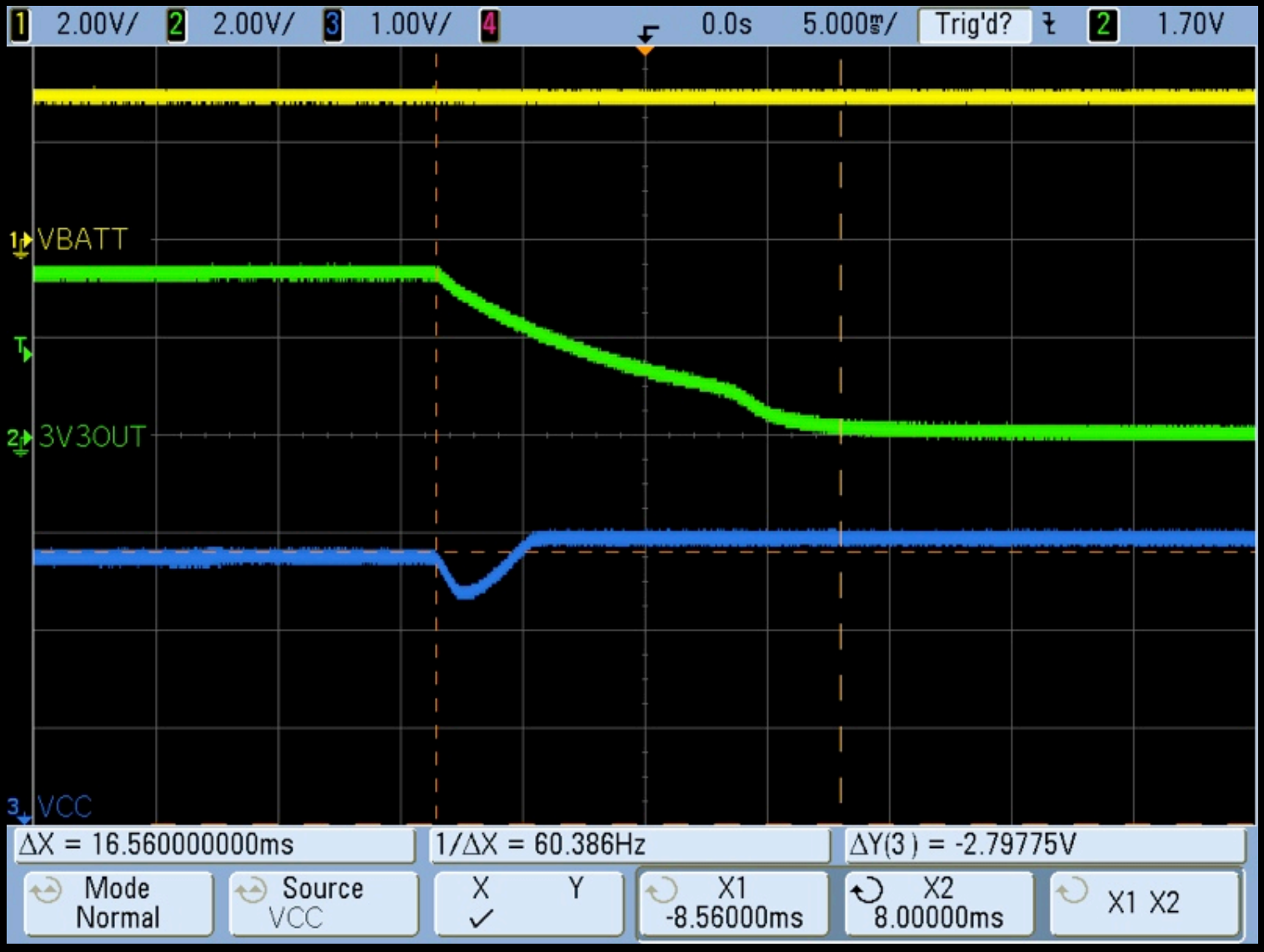
Seamless Power Switching 2

- Battery to USB

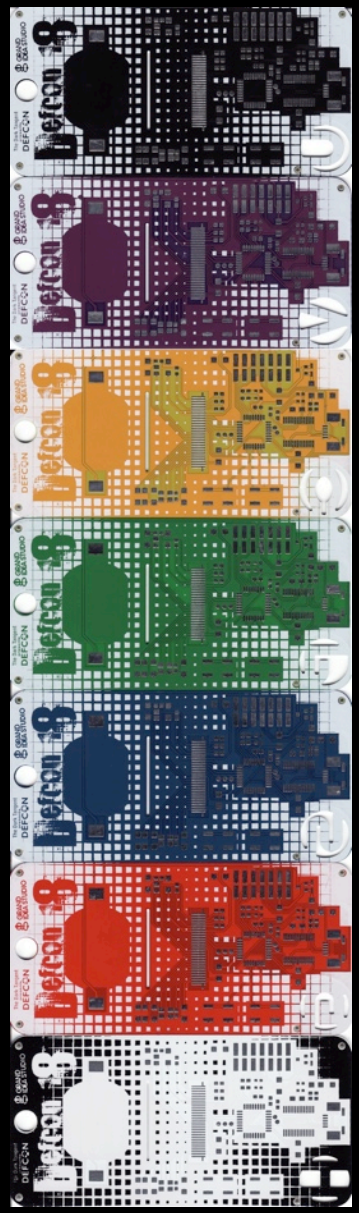


Seamless Power Switching 3

- USB to Battery



Total Badge Types



Human = 7000

Speaker = 200

Goon = 200

Press = 180

Vendor = 100

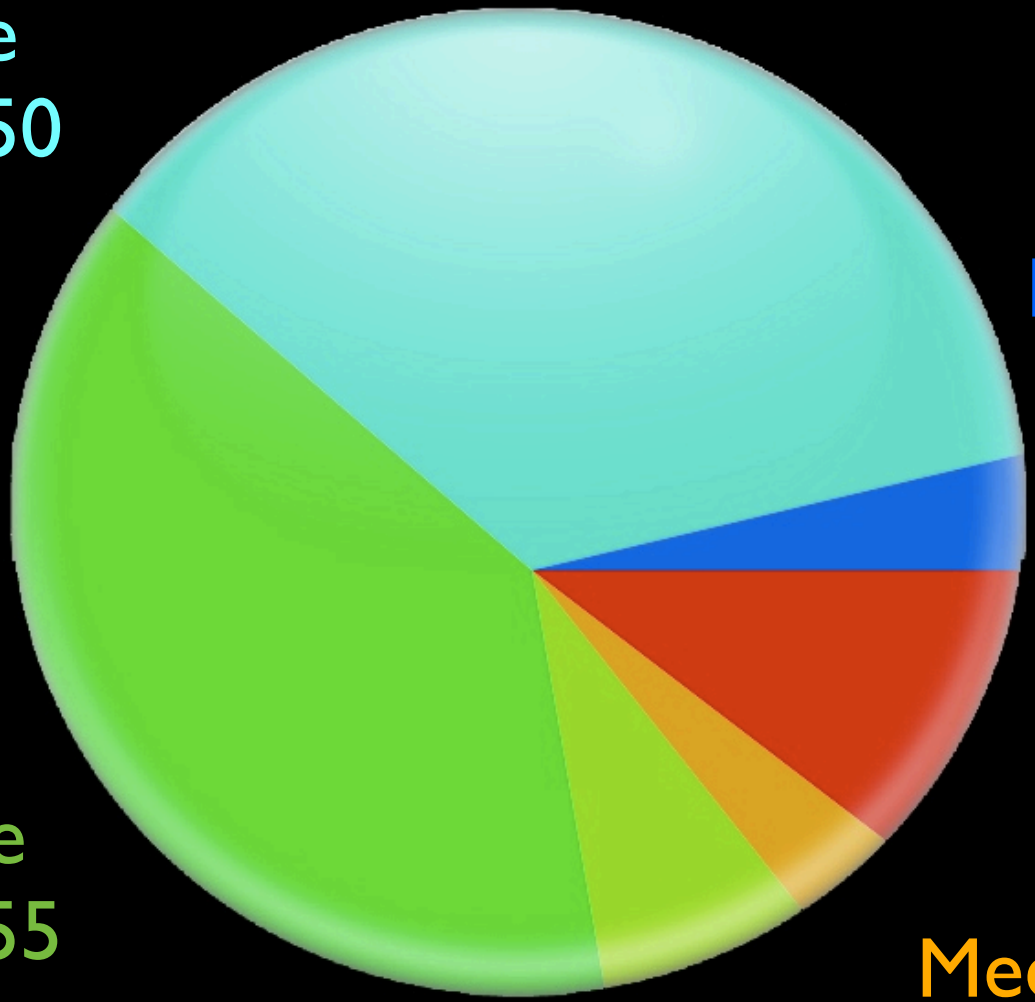
Contest = 70

Uber = 30

Total = 7780

Collect them all!@#

Time flies, but who's counting?



Firmware
34.6% 51:50

Documentation
3.8% 5:45

Research
10.7% 16:00

Hardware
39.3% 58:55

Meetings
3.9% 5:55

Admin
7.7% 11:35

TOTAL: 150 hours

Badge Hacking Contest

Now w/ Black Badge status!

Previous results at
www.grandideastudio.com/
portfolio/defcon-1x-badge/

x = 4, 5, 6, 7

Badge Hacking Contest HQ @
Hardware Hacking Village

Submit your entry to
Kingpin starting at 2pm
Sunday in the HHV

Complete schematic, source code, tools, etc. on DEFCON CD


DEFCON
THE END!
JOE@GRANDIDEASTUDIO.COM