

# Retrosshield 8008 for Arduino Mega/Teensy

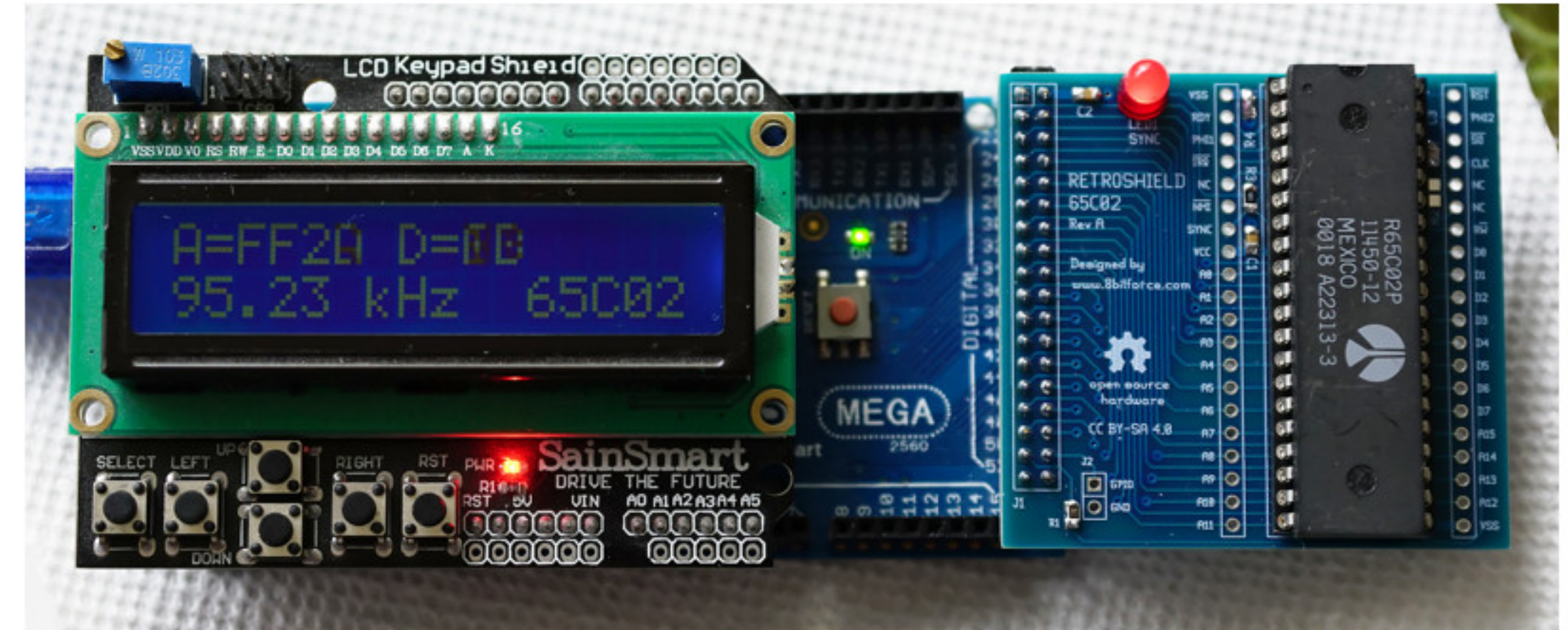
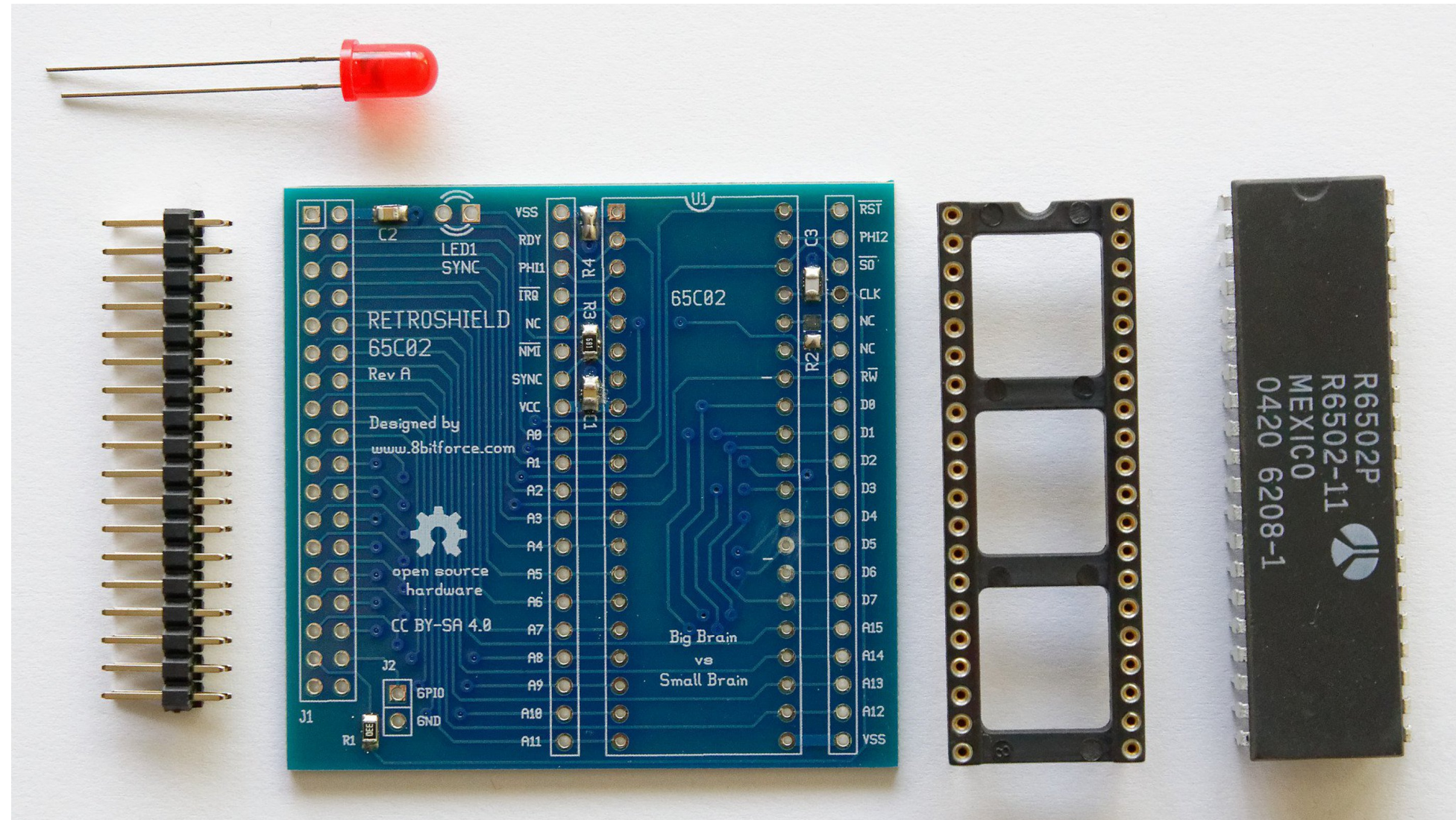
*Breadboarding in software...*

Erturk Kocalar  
erturkk@8bitforce.com

<https://gitlab.com/8bitforce>

29 MAR 2025

# Breadboarding in Software



## RetroShield for Arduino Mega/Teensy

Microprocessor daughter card that plugs into Arduino Mega/Teensy.

Real processor executes code while Arduino/Mega emulate system hw.

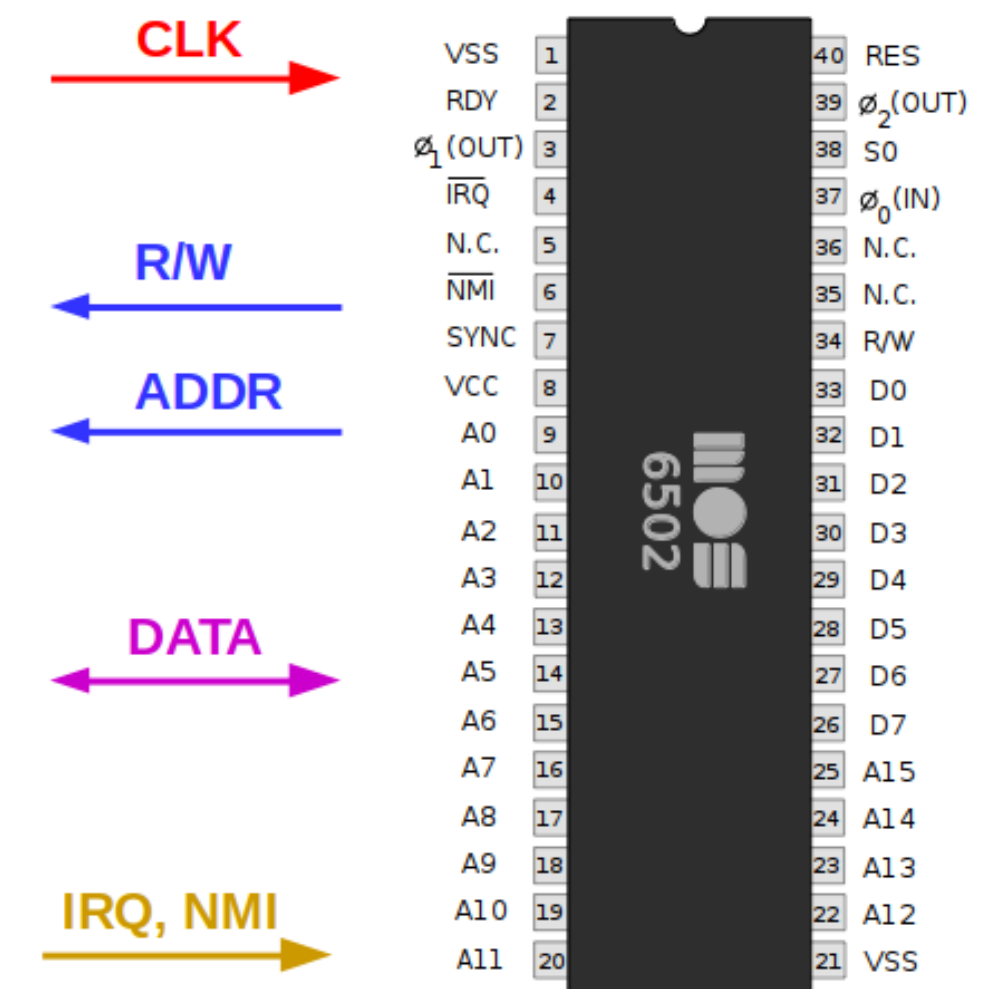
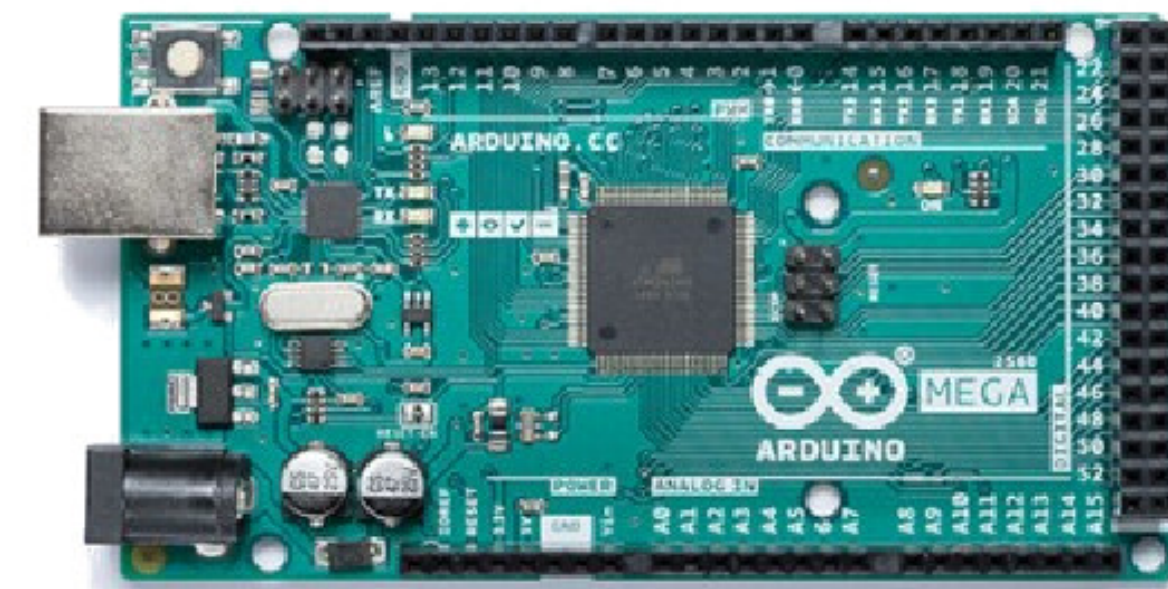
Switching system hardware is as easy as uploading new Arduino code.

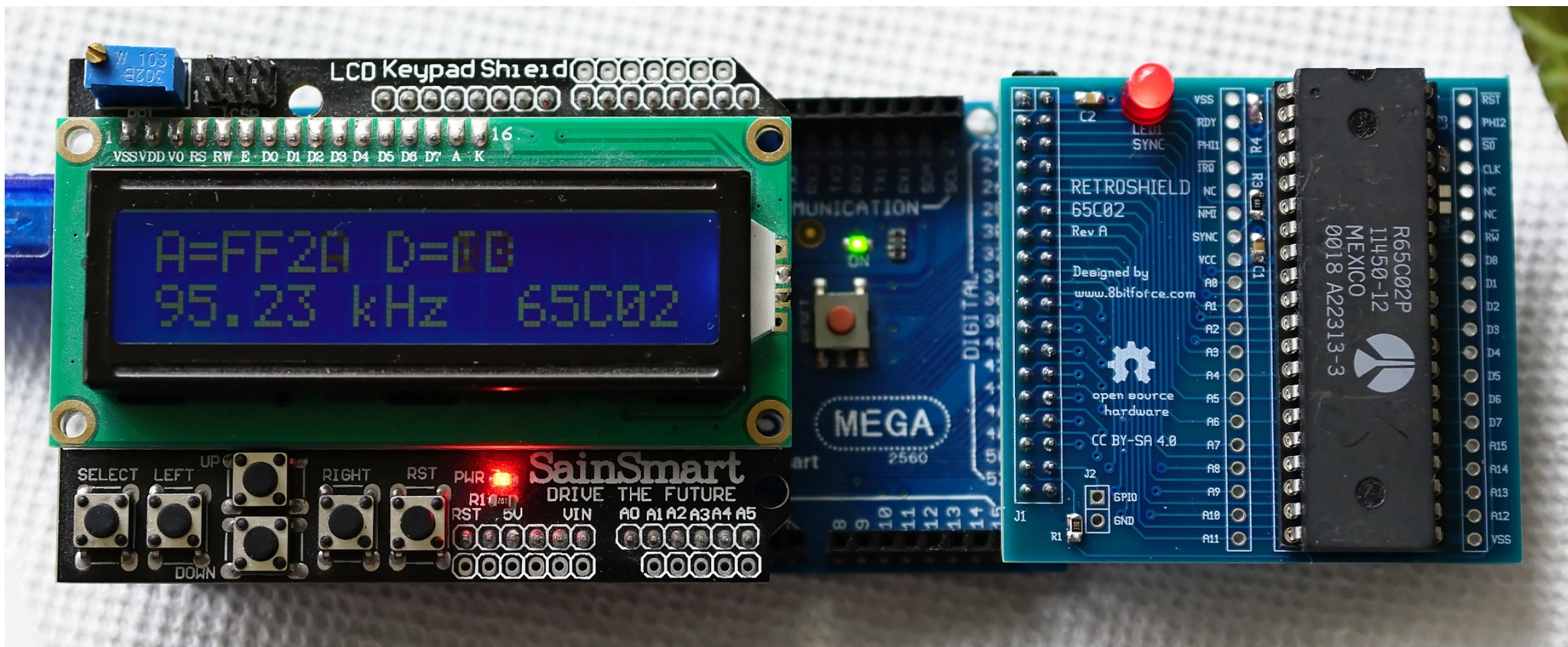
Software development done in Arduino IDE, with Arduino shields & libraries.

Open source hardware/software.

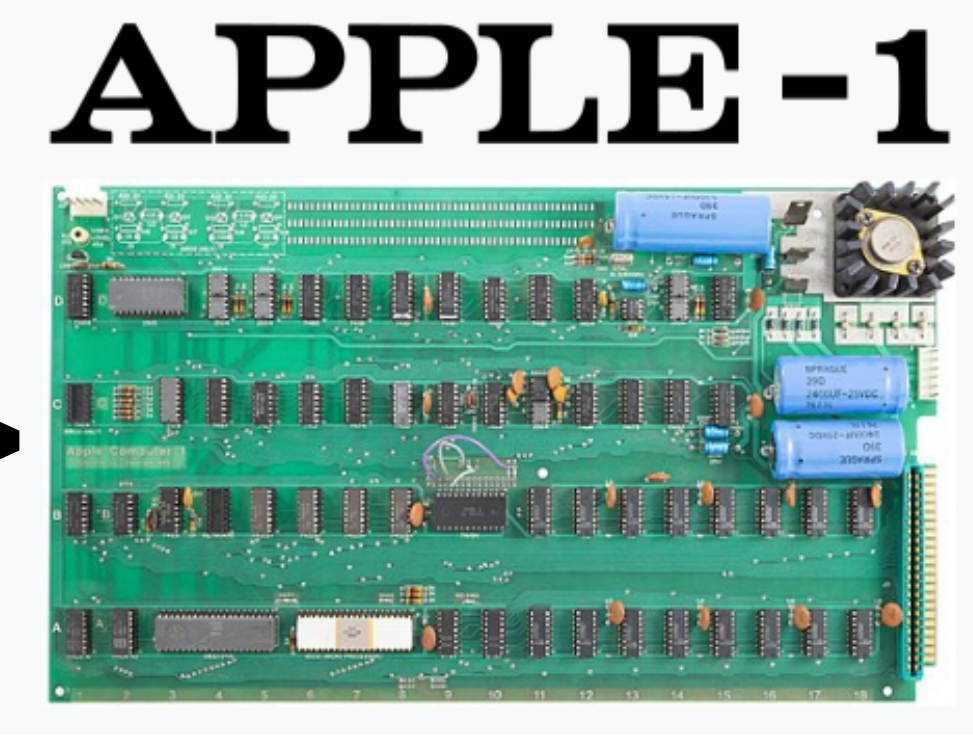
## CPU Daughtercards already released:

- 1Bit: 14500B
- 4Bit: 4004, 4040
- 8Bit: 8008, 8031, 8080, 8085, 8088
- 8Bit: 6502, 6803, 6809
- 8Bit: 1802, 2650, Z80, INS8060(SC/MP II)
- 12Bit: HD6120 (PDP8)
- 16Bit: 68008

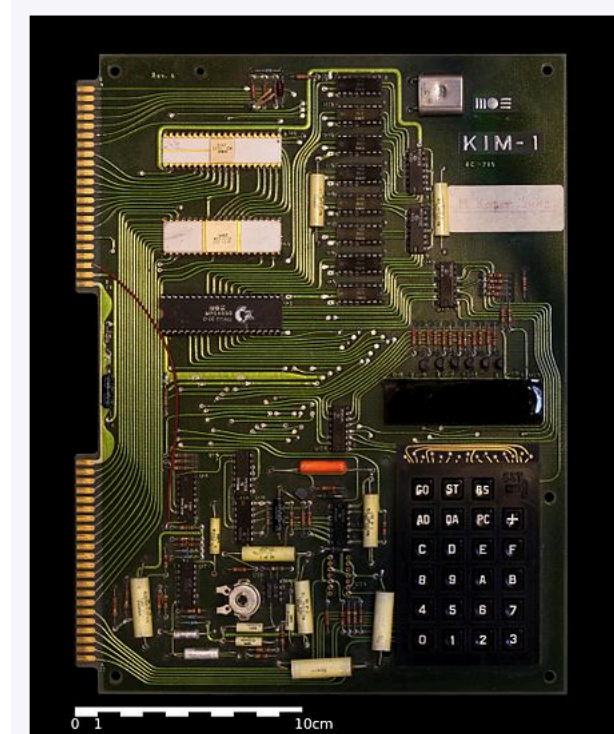




⇒



KIM-1



Credit Wikipedia

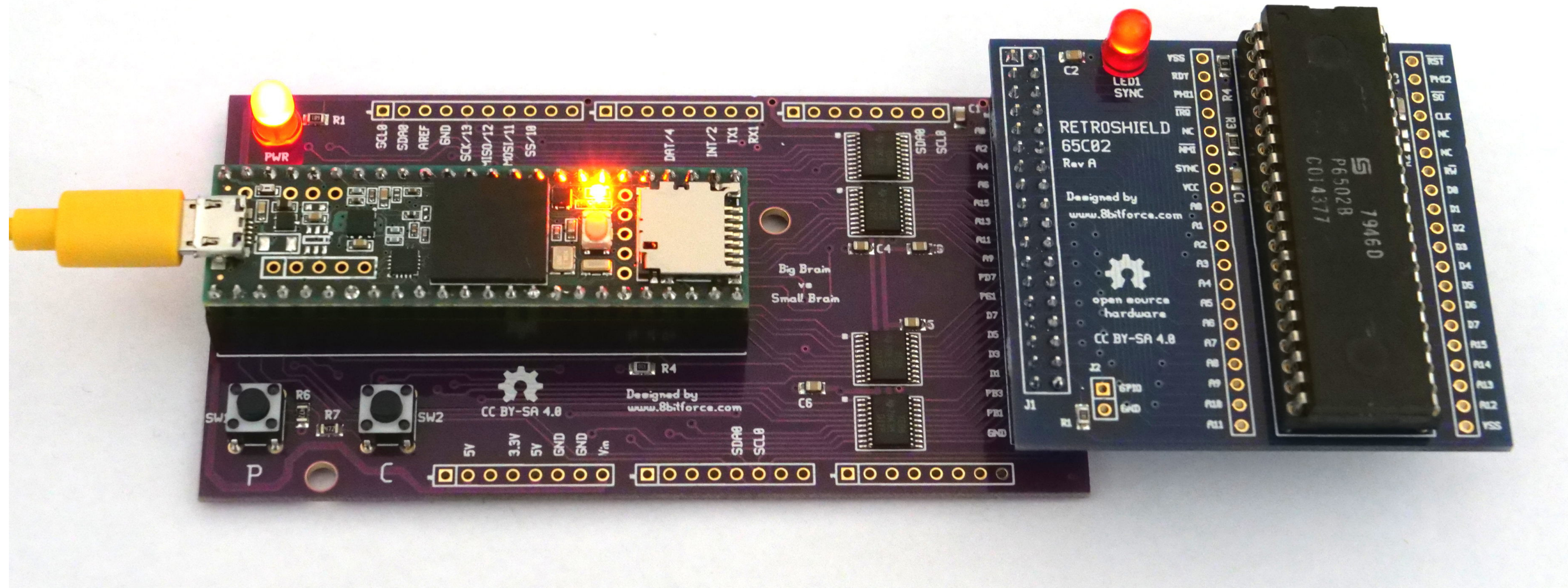
```

k65c02_apple1 | Arduino IDE 2.3.4
Teensy 4.1
k65c02_apple1.ino  README.md  pins2_arduino.h
512 #endif
513
514 // ROMs (Monitor + Basic + ACI)
515 #define ROM_START 0xFF00
516 #define ROM_END 0xFFFF
517 #define BASIC_START 0xE000
518 #define BASIC_END 0xEFFF
519 // added Lorenz Born
520 #define ACI_START 0xC100
521 #define ACI_END 0xC1FF
522
523 ///////////////////////////////////////////////////////////////////
524 // Woz Monitor Code
525 ///////////////////////////////////////////////////////////////////
526 // static const unsigned char
527 PROGMEM const unsigned char rom_bin[] = {
528 // static const unsigned char rom_bin[] = {
529 0xd8, 0x58, 0xa0, 0x7f, 0x8c, 0x12, 0xd0, 0xa9, 0xa7, 0x8d, 0x11, 0xd0,
530 0x8d, 0x13, 0xd0, 0xc9, 0xdf, 0xf0, 0x13, 0xc9, 0x9b, 0xf0, 0x03, 0xc8,
531 0x10, 0x0f, 0xa9, 0xdc, 0x20, 0xef, 0xff, 0xa9, 0x8d, 0x20, 0xef, 0xff,
532 0xa0, 0x01, 0x88, 0x30, 0xf6, 0xad, 0x11, 0xd0, 0x10, 0xfb, 0xad, 0x10,
533 0xd0, 0x99, 0x00, 0x02, 0x20, 0xef, 0xff, 0xc9, 0x8d, 0xd0, 0xd4, 0xa0,
534 0xff, 0xa9, 0x00, 0xaa, 0x0a, 0x85, 0x2b, 0xc8, 0xb9, 0x00, 0x02, 0xc9,
535 0x8d, 0xf0, 0xd4, 0xc9, 0xae, 0x90, 0xf4, 0xf0, 0xf0, 0xc9, 0xba, 0xf0,
536 0xeb, 0xc9, 0xd2, 0xf0, 0x3b, 0x86, 0x28, 0x86, 0x29, 0x84, 0x2a, 0xb9,
537 0x00, 0x02, 0x49, 0xb0, 0xc9, 0x0a, 0x90, 0x06, 0x69, 0x88, 0xc9, 0xfa,
538 0x90, 0x11, 0x0a, 0x0a, 0x0a, 0x0a, 0xa2, 0x04, 0x0a, 0x26, 0x28, 0x26,
539 0x29, 0xca, 0xd0, 0xf8, 0xc8, 0xd0, 0xe0, 0xc4, 0x2a, 0xf0, 0x97, 0x24,
540 0x2b, 0x50, 0x10, 0xa5, 0x28, 0x81, 0x26, 0xe6, 0x26, 0xd0, 0xb5, 0xe6,
541 0x27, 0x4c, 0x44, 0xff, 0x6c, 0x24, 0x00, 0x30, 0x2b, 0xa2, 0x02, 0xb5,
542 0x27, 0x95, 0x25, 0x95, 0x23, 0xca, 0xd0, 0xf7, 0xd0, 0x14, 0xa9, 0x8d,
543 0x20, 0xef, 0xff, 0xa5, 0x25, 0x20, 0xdc, 0xff, 0xa5, 0x24, 0x20, 0xdc,
544 0xff, 0xa9, 0xba, 0x20, 0xef, 0xff, 0xa9, 0xa0, 0x20, 0xef, 0xff, 0xa1,
545 0x24, 0x20, 0xdc, 0xff, 0x86, 0x2b, 0xa5, 0x24, 0xc5, 0x28, 0xa5, 0x25,
546 0xe5, 0x29, 0xb0, 0xc1, 0xe6, 0x24, 0xd0, 0x02, 0xe6, 0x25, 0xa5, 0x24,
547 0x29, 0x07, 0x10, 0xc8, 0x48, 0x4a, 0x4a, 0x4a, 0x20, 0xe5, 0xff,
548 0x68, 0x29, 0x0f, 0x09, 0xb0, 0xc9, 0xba, 0x90, 0x02, 0x69, 0x06, 0x2c,
549 0x12, 0xd0, 0x30, 0xfb, 0x8d, 0x12, 0xd0, 0x60, 0x00, 0x00, 0x00, 0x0f,
550 0x00, 0xff, 0x00, 0x00
551 };
552
553 ///////////////////////////////////////////////////////////////////

```

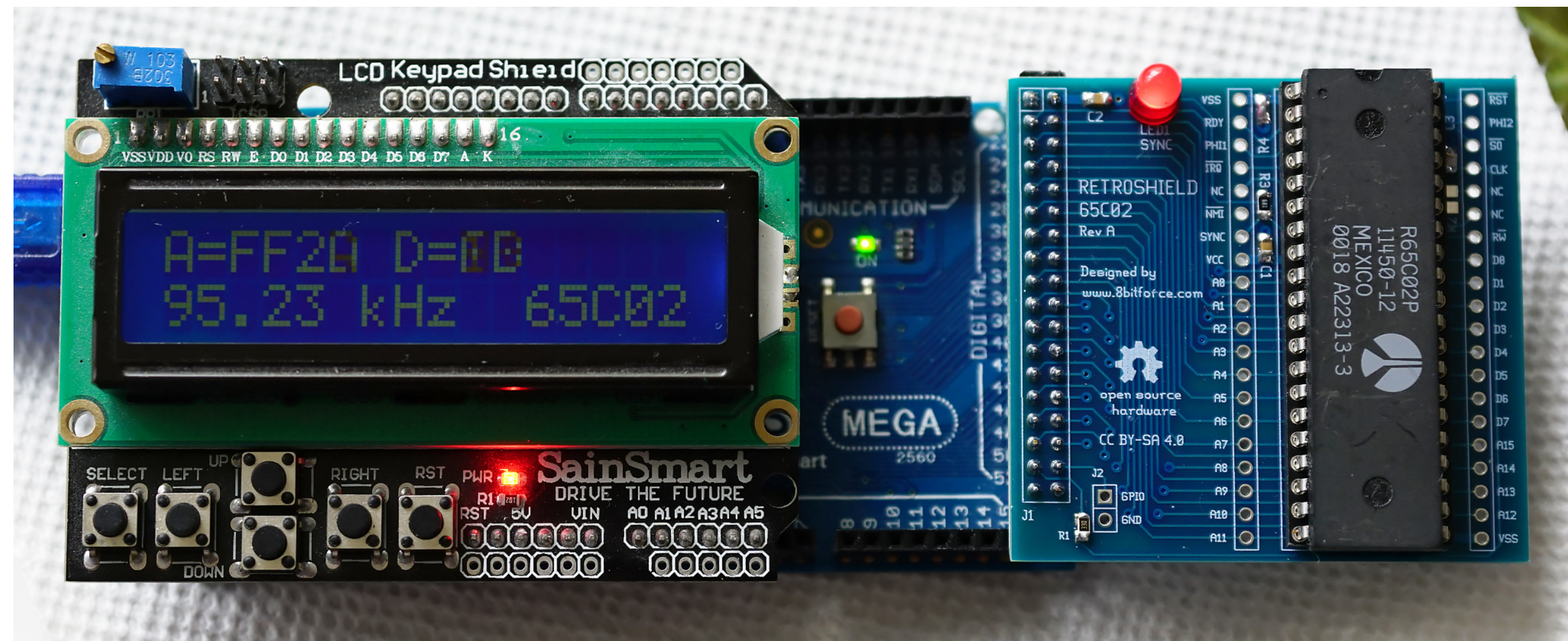
- Experience old computer systems in a cheap & easy way.
- Replace ROM contents easily by changing C code.
- Keep historical/valuable software alive.
- Design your own custom computer using software.
- Learn low level hardware interfacing w/ microprocessors.

# Two platforms



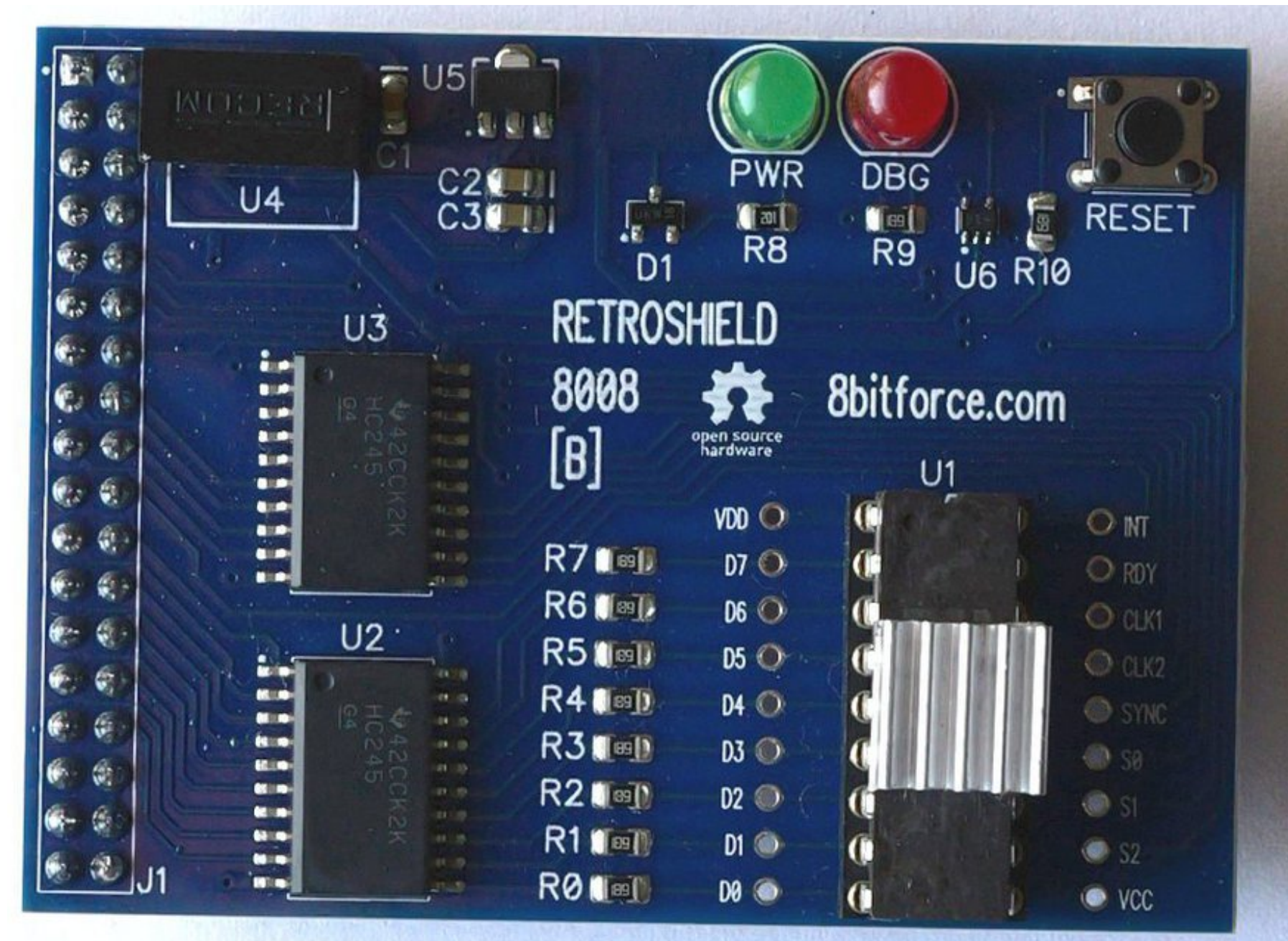
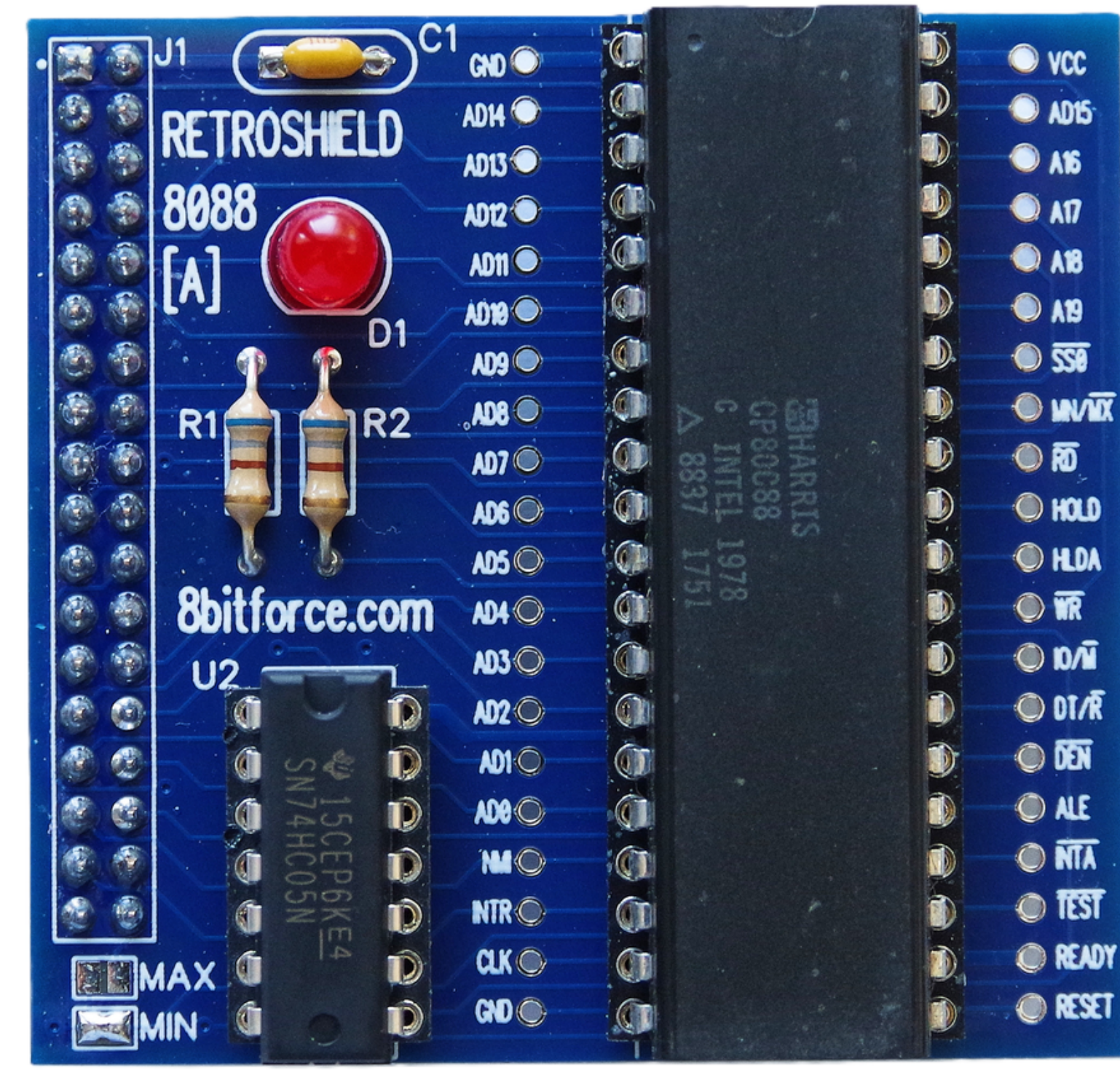
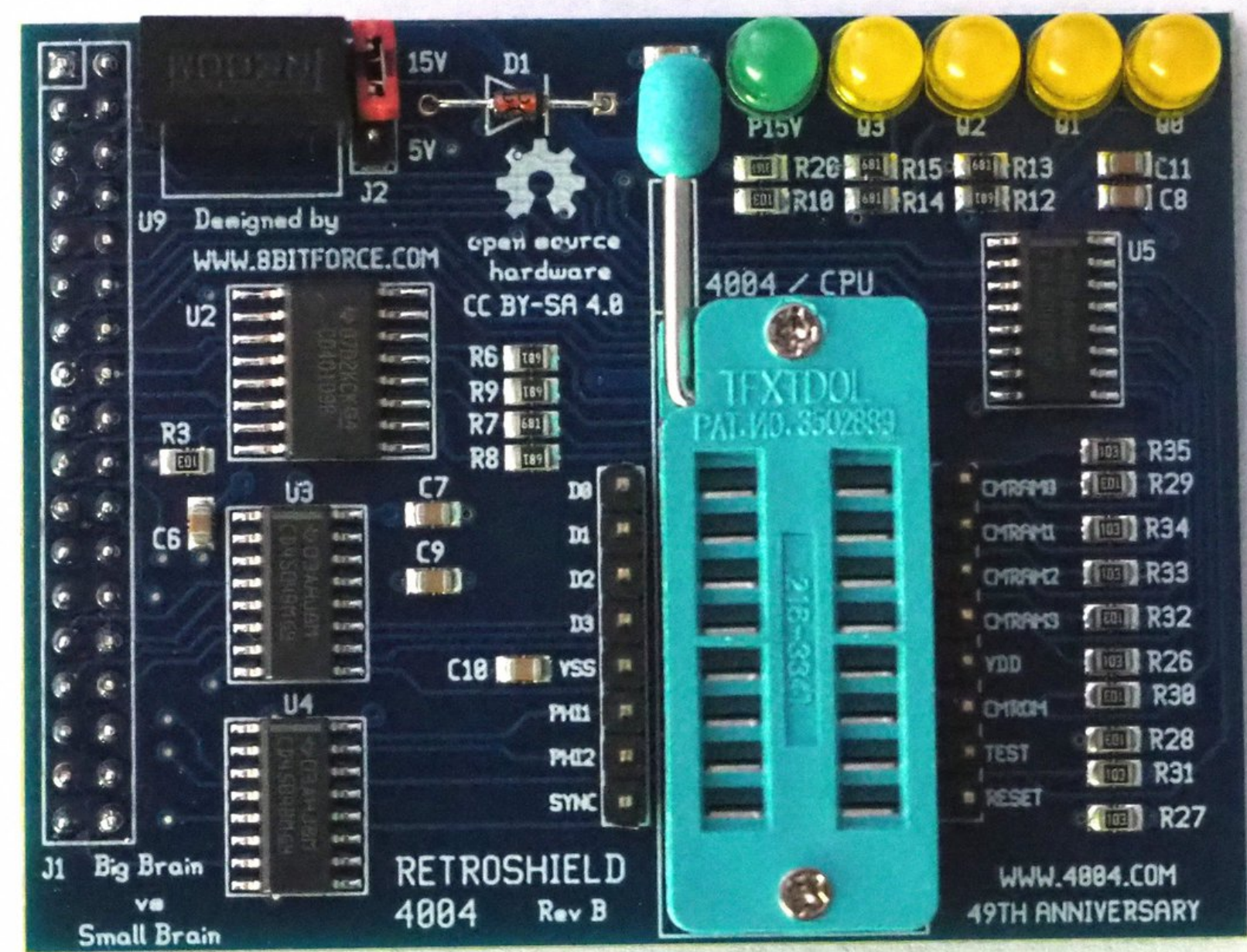
## Teensy 4.1 (600MHz) Fast Platform:

- Microprocessor can run at full speed, i.e. 1.00+ MHz
- **RAM : 1MB (Teensy 4.1), 256 KB (Teensy 3.5/3.6)**
- ROM : >512 KB
- **Onboard 3.3V $\Leftrightarrow$ 5V level shifting**
- microSD slot (can be used to emulate disk drives)
- Peripherals: UART, PIA, Timers, etc. (emulated by Teensy)

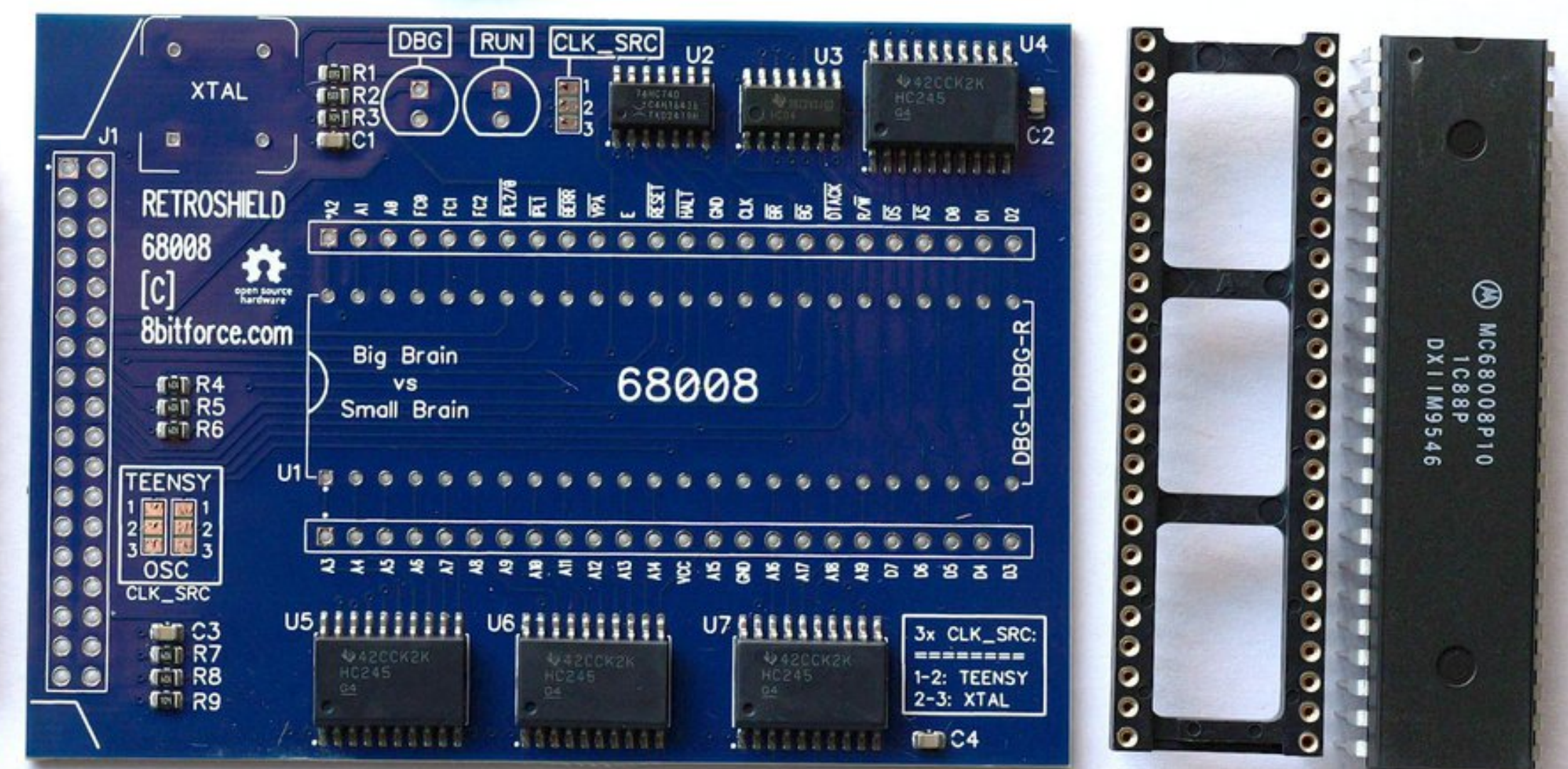
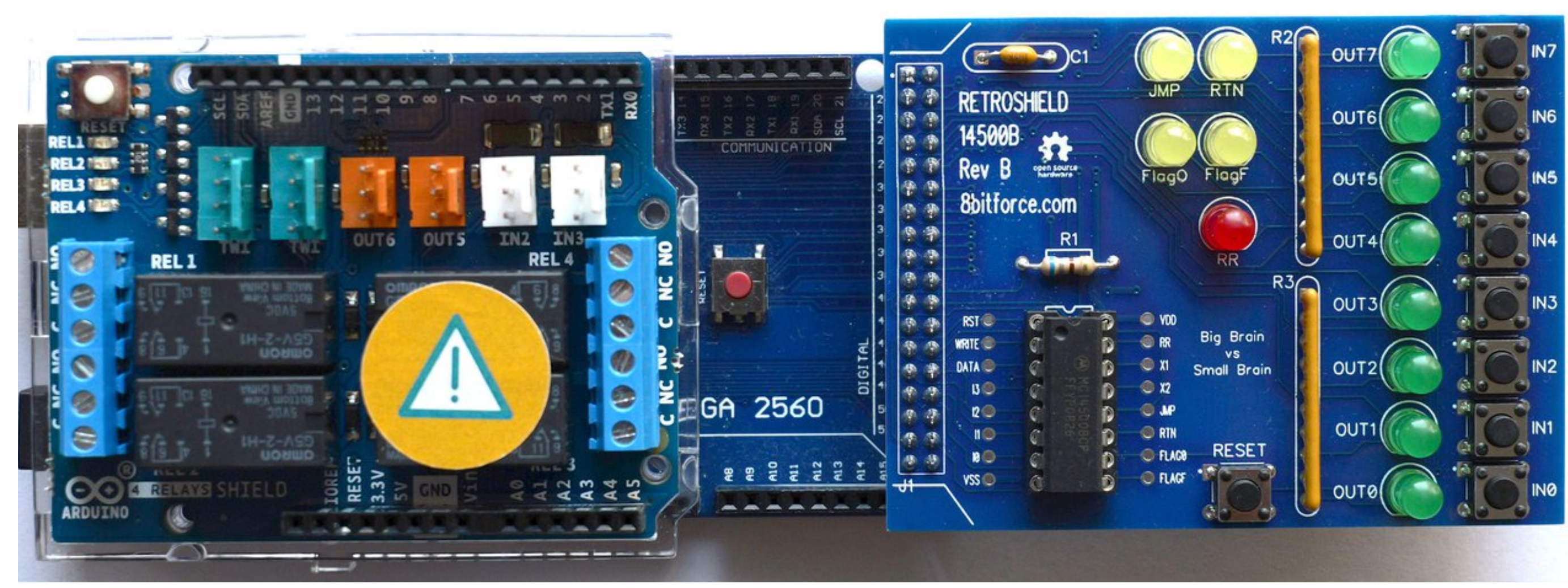


## Arduino Mega (16MHz) Cheap Platform:

- Microprocessor runs at about 100~200kHz up to 400kHz.
- ROM : >200 KB
- **RAM : ~6 KB (2KB of 8KB reserved for Arduino stack/heap)**
- Peripherals: UART, PIA, Timers, etc. (emulated by Arduino)
- **Arduino Shields can be used to add new features.**

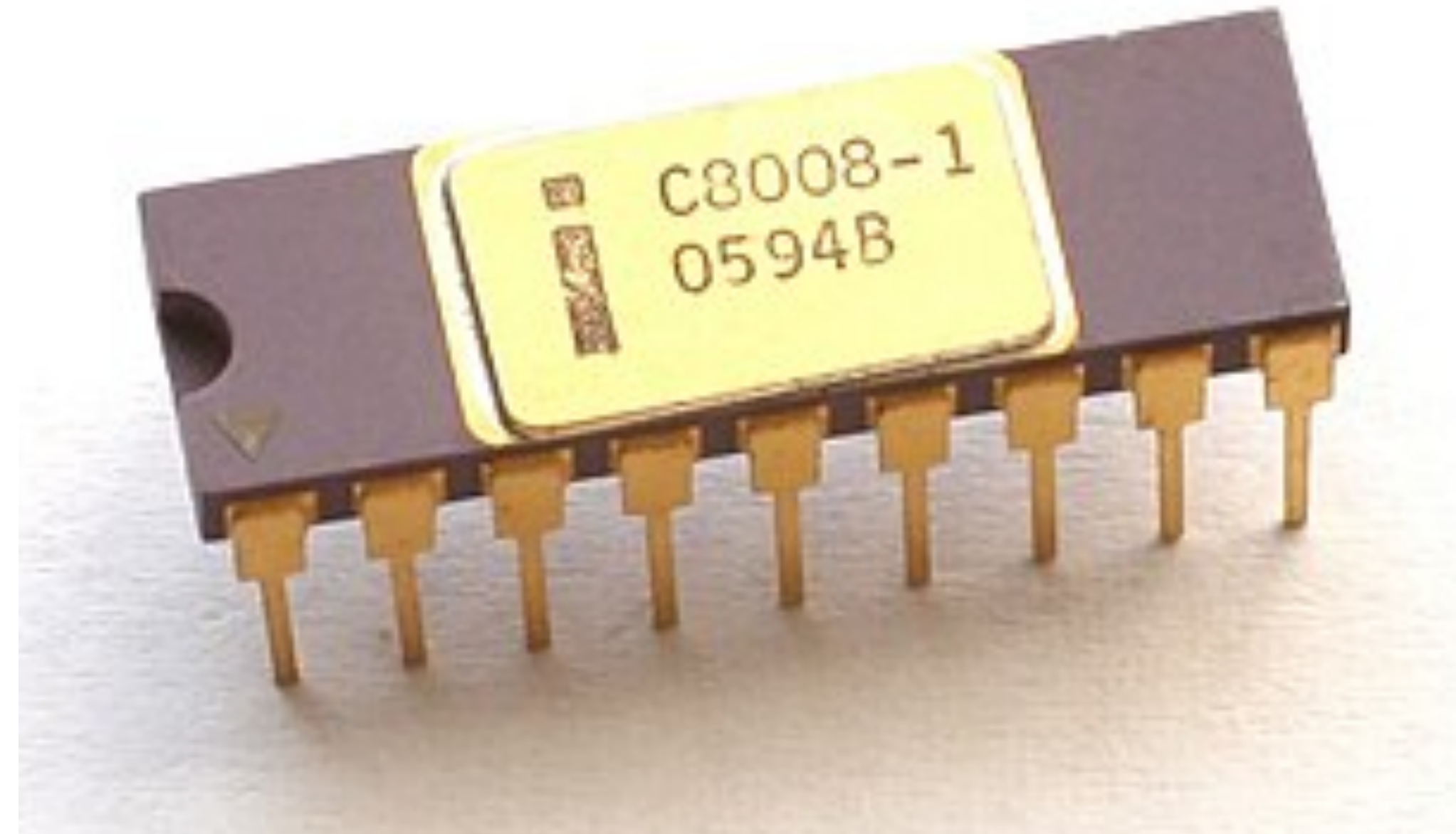


*Retroshields come in many shapes and forms...*

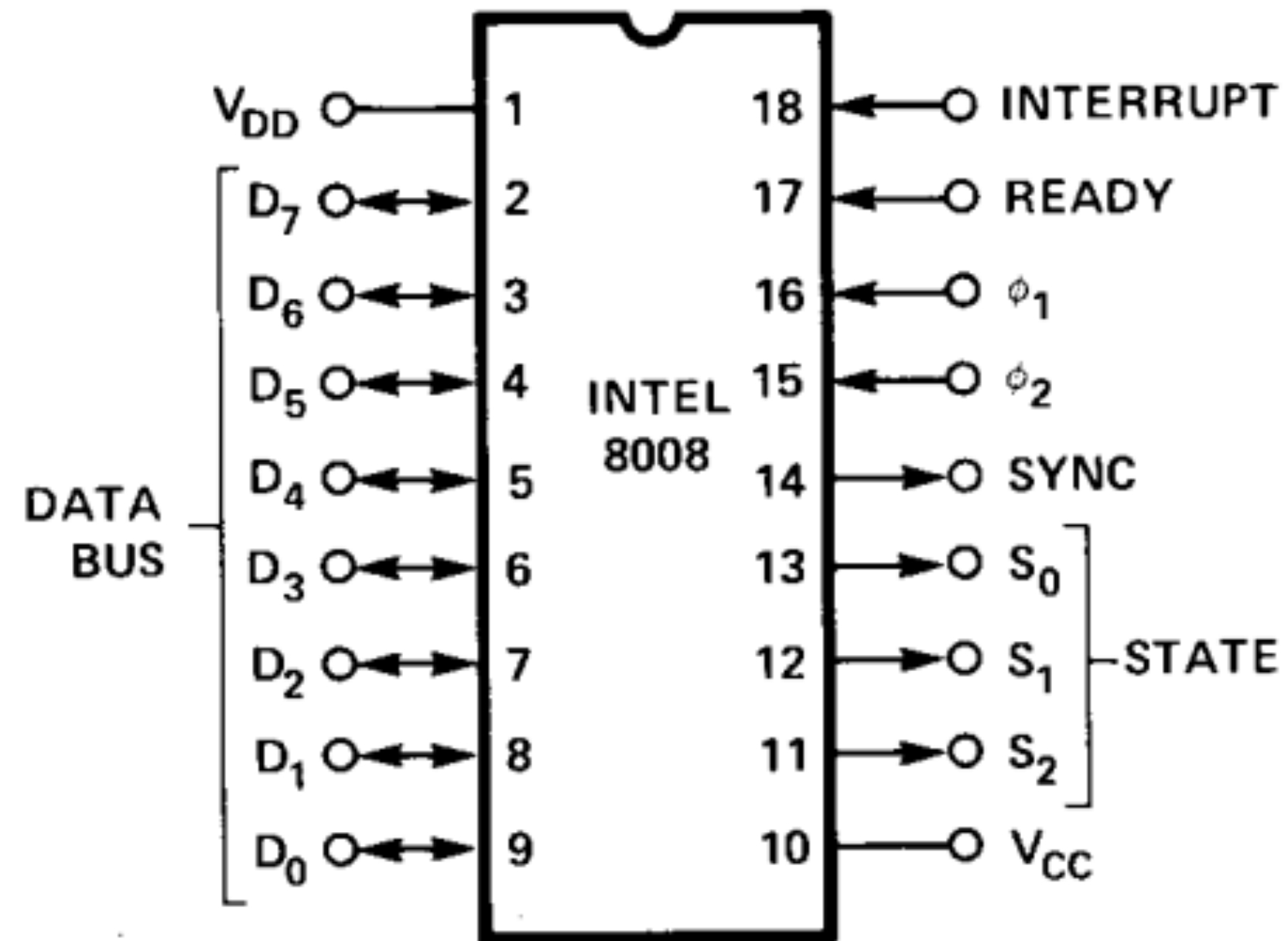


# RetroShield 8008

World's first 8-bit microprocessor



# RetroShield 8008



- Two power supplies: +5V and -9V. No GND pin.
- INTERRUPT pin requires externally feeding one instruction for interrupt handler, usually RST.
- no RESET input. on power-up, interrupt pin used to feed the start-up instruction, i.e. RST (Restart) or JMP EXEC

## Intel 8008 registers

1<sub>3</sub> 1<sub>2</sub> 1<sub>1</sub> 1<sub>0</sub> 0<sub>9</sub> 0<sub>8</sub> 0<sub>7</sub> 0<sub>6</sub> 0<sub>5</sub> 0<sub>4</sub> 0<sub>3</sub> 0<sub>2</sub> 0<sub>1</sub> 0<sub>0</sub> (bit position)

### Main registers

	A	Accumulator
	B	B register
	C	C register
	D	D register
	E	E register
	H	H register (indirect)
	L	L register (indirect)

### Program counter

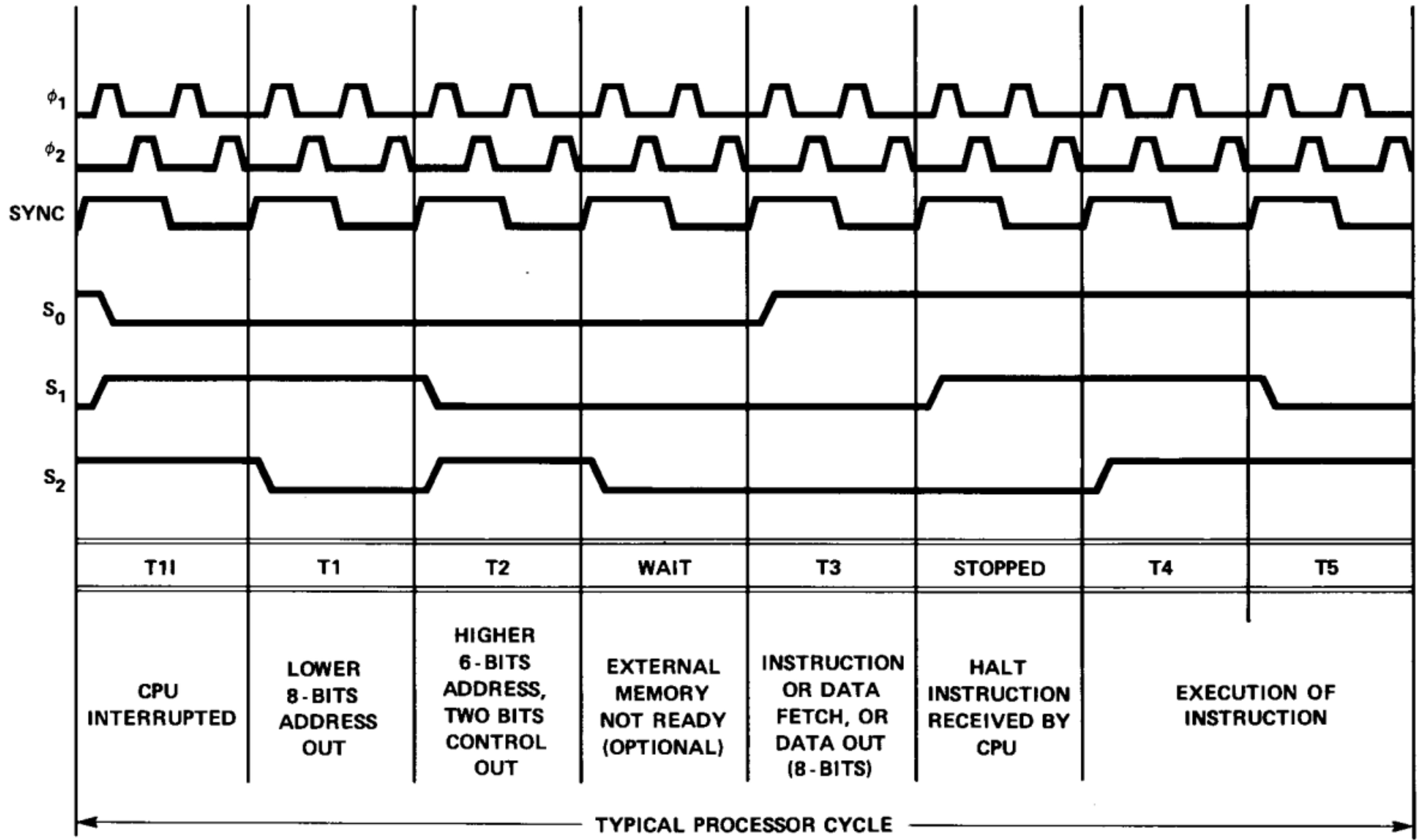
PC	Program Counter
----	-----------------

### Push-down address call stack

AS	Call level 1
AS	Call level 2
AS	Call level 3
AS	Call level 4
AS	Call level 5
AS	Call level 6
AS	Call level 7

### Flags

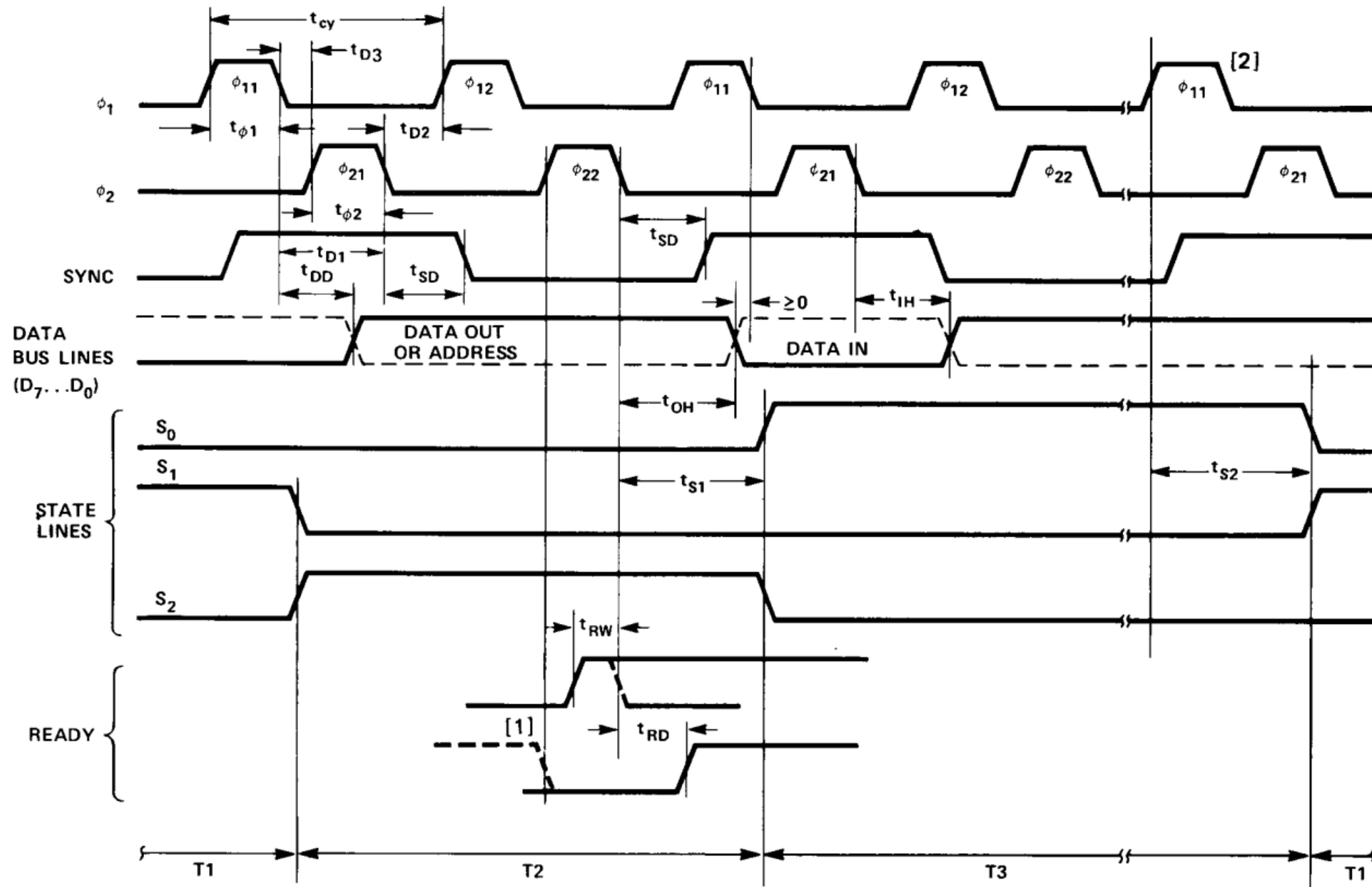
C	P	Z	S	Flags <sup>[12][a]</sup>
---	---	---	---	--------------------------



TYPICAL PROCESSOR CYCLE INCLUDES T1, T2, T3, T4, T5

Similar to 4004

# TIMING DIAGRAM



- [1] READY line must be at "0" prior to  $\phi_{22}$  of T2 to guarantee entry into the WAIT state.
- [2] INTERRUPT line must not change levels within 200ns (max.) of the falling edge of  $\phi_1$ .

## A.C. CHARACTERISTICS

$T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ;  $V_{CC} = +5V \pm 5\%$ ,  $V_{DD} = -9V \pm 5\%$ . All measurements are referenced to 1.5V levels.

SYMBOL	PARAMETER	8008		8008-1		UNIT	TEST CONDITIONS
		LIMITS		LIMITS			
		MIN.	MAX.	MIN.	MAX.		
$t_{CY}$	CLOCK PERIOD	2	3	1.25	3	$\mu\text{s}$	$t_R, t_F = 50\text{ns}$
$t_{R, t_F}$	CLOCK RISE AND FALL TIMES		50		50	ns	
$t_{\phi 1}$	PULSE WIDTH OF $\phi_1$	.70		.35		$\mu\text{s}$	
$t_{\phi 2}$	PULSE WIDTH OF $\phi_2$	.55		.35		$\mu\text{s}$	
$t_{D1}$	CLOCK DELAY FROM FALLING EDGE OF $\phi_1$ TO FALLING EDGE OF $\phi_2$	.90	1.1		1.1	$\mu\text{s}$	
$t_{D2}$	CLOCK DELAY FROM $\phi_2$ TO $\phi_1$	.40		.35		$\mu\text{s}$	
$t_{D3}$	CLOCK DELAY FROM $\phi_1$ TO $\phi_2$	.20		.20		$\mu\text{s}$	
$t_{DD}$	DATA OUT DELAY		1.0		1.0	$\mu\text{s}$	$C_L = 100\text{pF}$
$t_{OH}$	HOLD TIME FOR DATA OUT	.10		.10		$\mu\text{s}$	
$t_{IH}$	HOLD TIME FOR DATA IN	[1]		[1]		$\mu\text{s}$	
$t_{SD}$	SYNC OUT DELAY		.70		.70	$\mu\text{s}$	$C_L = 100\text{pF}$
$t_{S1}$	STATE OUT DELAY (ALL STATES EXCEPT T1 AND T1I) [2]		1.1		1.1	$\mu\text{s}$	$C_L = 100\text{pF}$
$t_{S2}$	STATE OUT DELAY (STATES T1 AND T1I)		1.0		1.0	$\mu\text{s}$	$C_L = 100\text{pF}$
$t_{RW}$	PULSE WIDTH OF READY DURING $\phi_{22}$ TO ENTER T3 STATE	.35		.35		$\mu\text{s}$	
$t_{RD}$	READY DELAY TO ENTER WAIT STATE	.20		.20		$\mu\text{s}$	

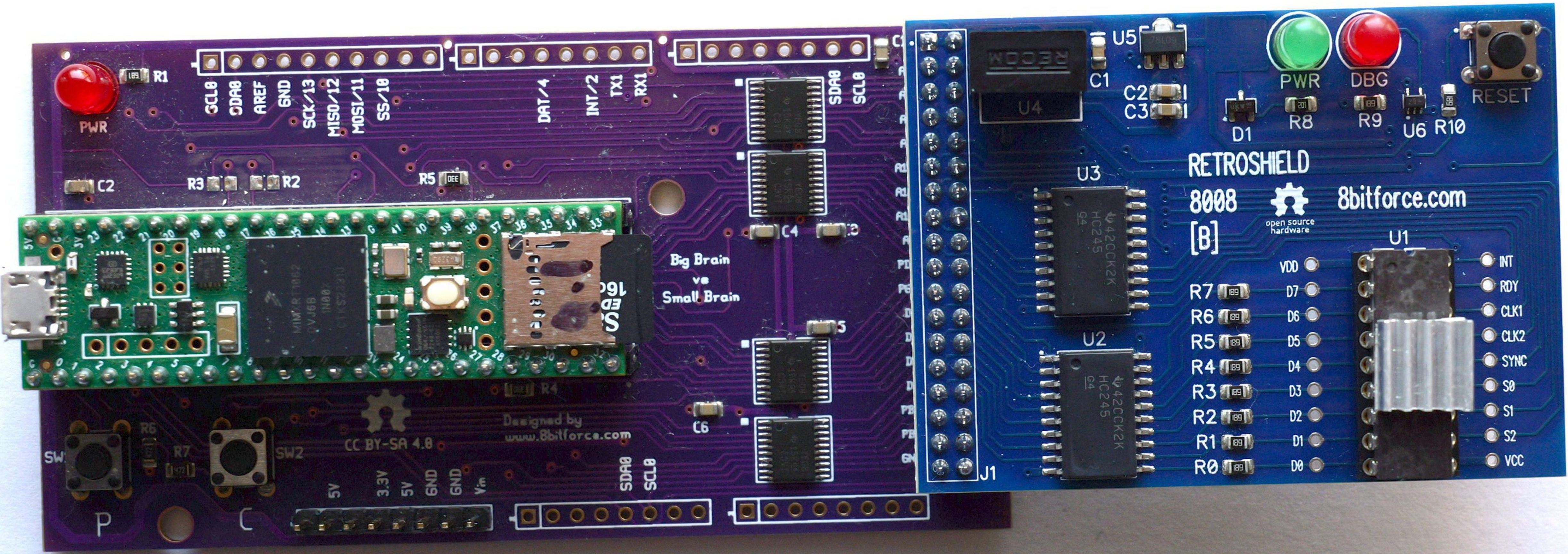
[1]  $t_{IH} \text{ MIN} \geq t_{SD}$

[2] If the INTERRUPT is not used, all states have the same output delay,  $t_{S1}$ .

Pay attention - clock must be between 333kHz ... 800kHz.

Reset button goes to Teensy to trigger interrupt/reset.

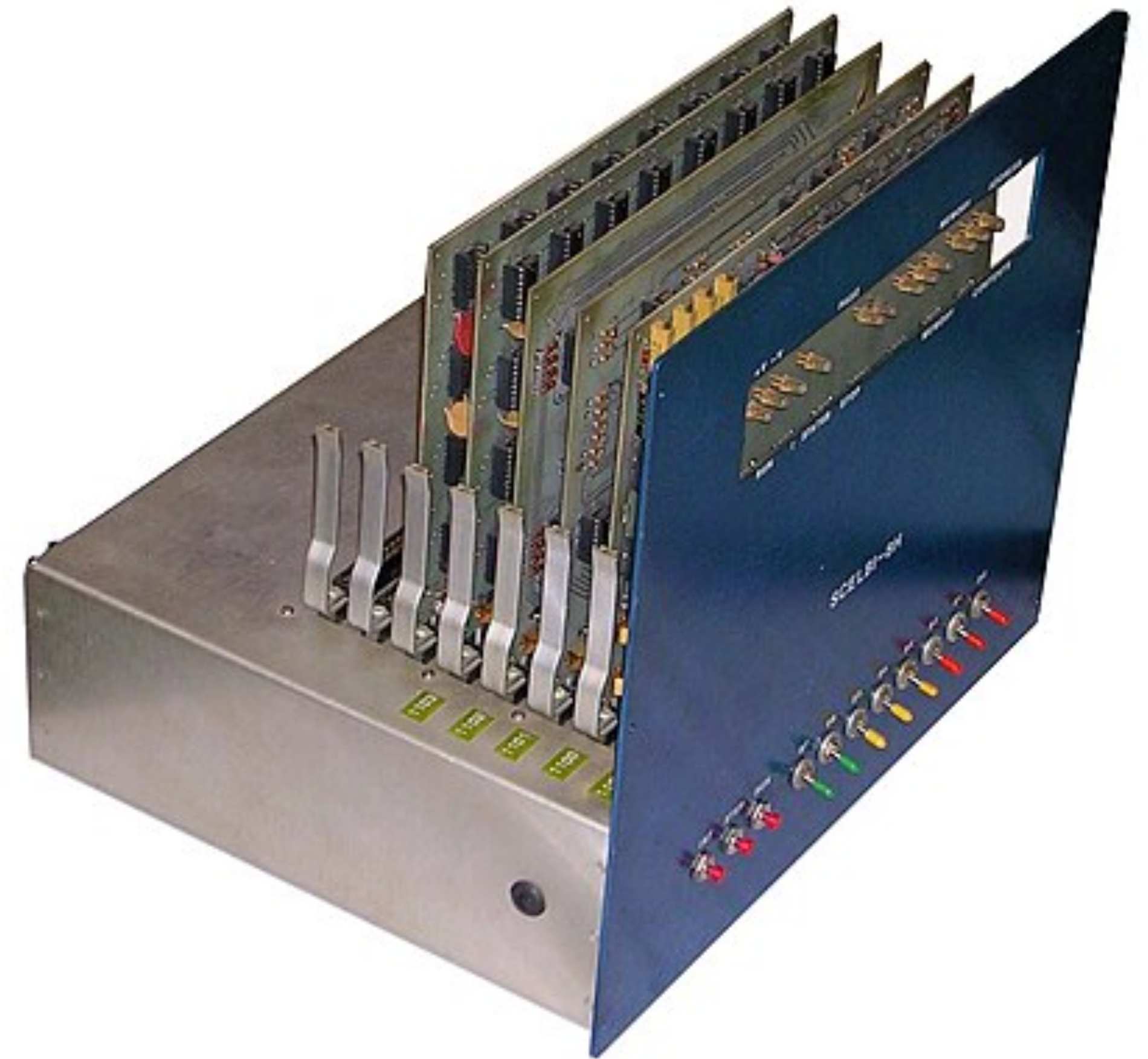
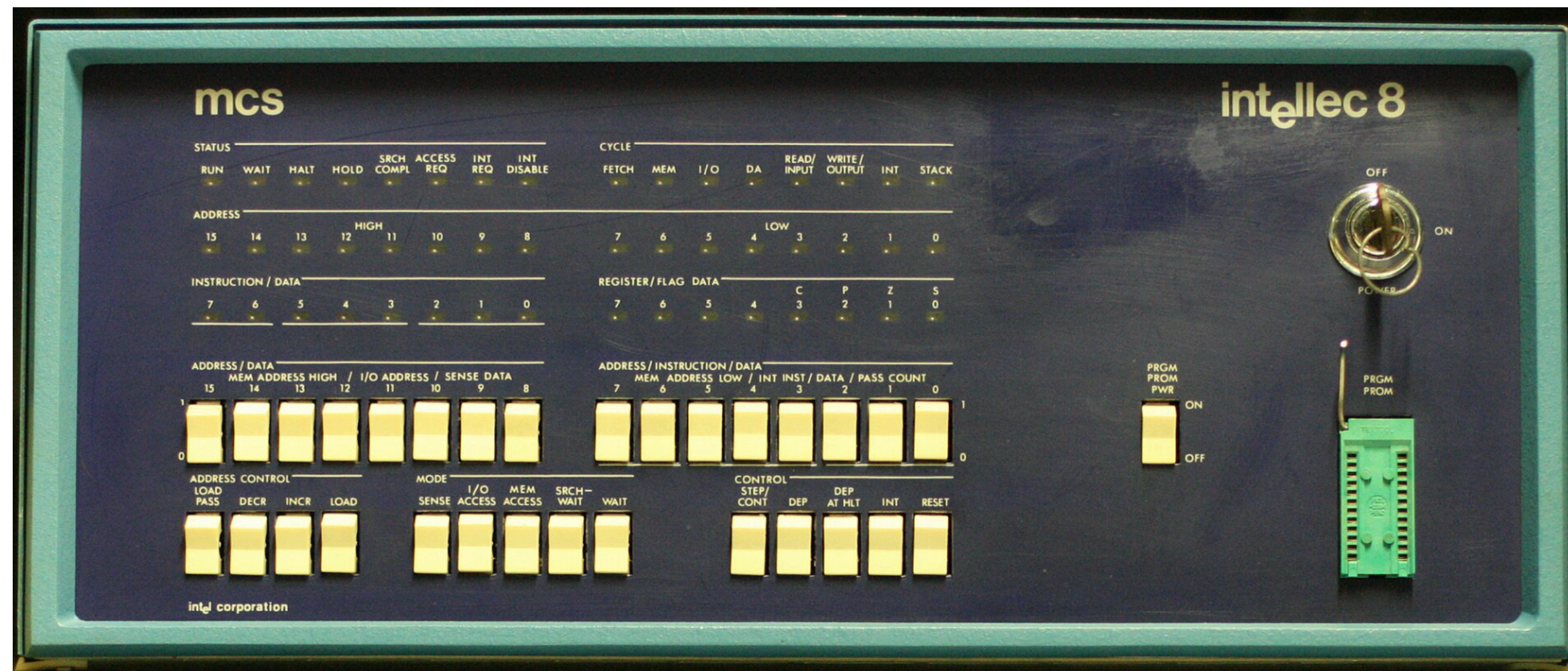
-9V Supply



Level Shifters

8008+Heatsink

# 8008 Uses



## INTELLEC 8/MOD 8

- ROM-Resident monitor (memory, tape/punch, tty)
- Editor/Assembler available on tape (?lost)
- PROM programmer

## SCELBI

- SCELBAL (SCientific ELementary BAsic Language)
- Book form w/ very well documented assembly
- Floating point support
- Math functions (Sin, Cos, Exp, Log, Atn)
- String and array DIM

t8008\_scelbal | Arduino IDE 2.3.4

Teensy 4.1

```

t8008_scelbal.ino  buttons.h  memorymap.h  portmap.h  setuphold.h  soft_uart.h  eeprom1.h  scelbal.h
191
192 // Initialize "hw" before cpu starts executing, such as
193 // filling memory or modify vectors or modifying rom functions.
194 void board_init()
195 {
196     // Clear RAM1
197     for (unsigned long s=0; s<sizeof(RAM1); s++)
198         RAM1[RAM1_START+s] = 0x00;
199
200     // Copy ROM1(16KB) to RAM1(64K) b/c some cpu_tick() prioritize RAM1 vs ROM1
201     for (word s=0; s<sizeof(ROM1); s++)
202         RAM1[RAM1_START+s] = ROM1[ROM1_START+s];
203
204     // 8008 comes out reset w/ RST0 instruction.
205     // Place JMP EXEC at $0000.
206     RAM1[0x0000] = 0104;
207     RAM1[0x0001] = 0266;
208     RAM1[0x0002] = 0010;
209
210
211 #if SCELBAL_USES_SOFT_UART
212     // sc1.asm uses soft-uart, so we'll use soft-uart too.
213     // slow.
214 #else
215     // Override CINP
216     RAM1[0x0043] = 0113;    // CINP:  INP INPORT
217     RAM1[0x0044] = 0240;    //          NDA
218     RAM1[0x0045] = 0120;    //          JFS CINP
219     RAM1[0x0046] = 0103;
220     RAM1[0x0047] = 0000;
221     RAM1[0x0048] = 0155;    //          OUT OUTPORT
222     RAM1[0x0049] = 0007;    //          RET
223     // Override CPRINT
224     RAM1[0x0088] = 0155;    // CPRINT: OUT OUTPORT
225     RAM1[0x0089] = 0007;    //          RET
226 #endif
227

```

Output

```

FLASH: code:39080, data:22472, headers:9100  free for files:8055812
RAM1: variables:88928, code:36384, padding:29152  free for local variables:369824
RAM2: variables:12416  free for malloc/new:511872

```

Ln 8, Col 1

board\_init() prepares RAM before 8008 starts:

0) Copies EEPROM to RAM

1) Add JMP EXEC instruction at \$0000.

2) Override UART functions for speed.

3) Preload BASIC program in memory.

```

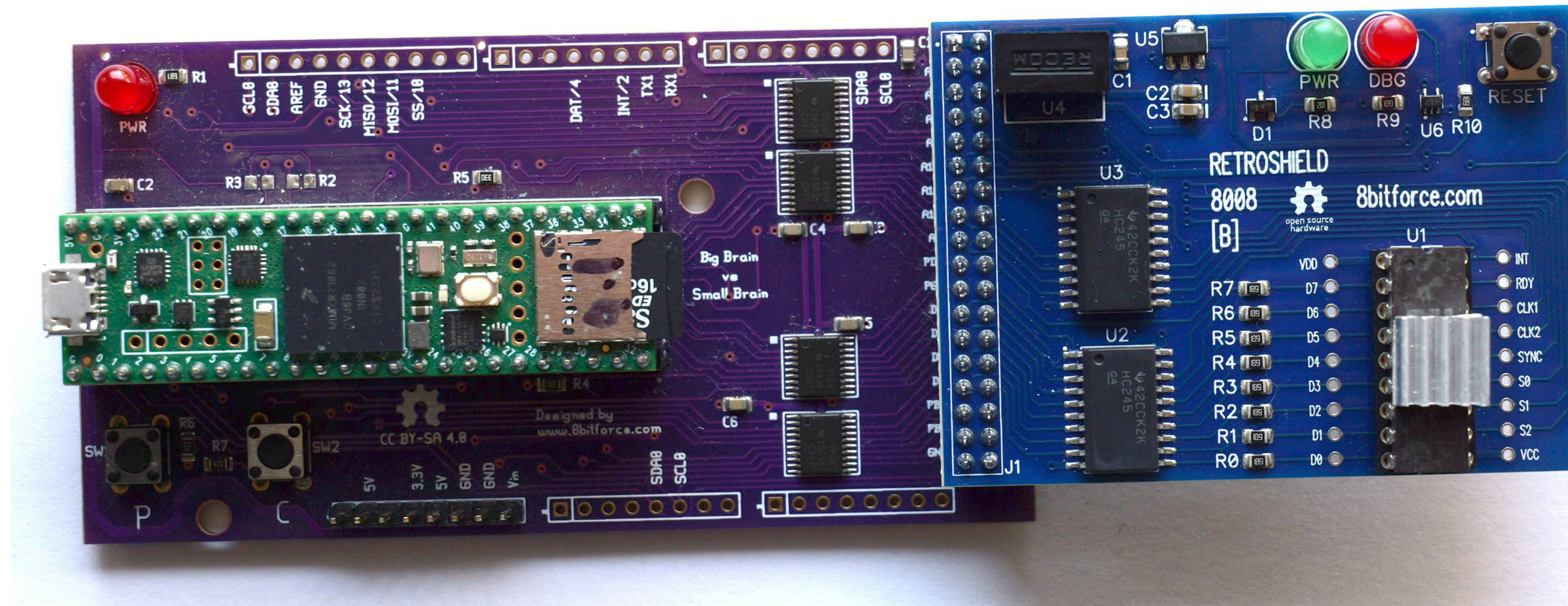
229     // Save a default program in memory.
230     // each line entry has i) length of line, ii) line it self.
231 #if (1)
232     word prog_addr = 0x1b00;
233     prog_addr = save_basic_line(prog_addr, "10 PRINT 'RADIUS'");
234     prog_addr = save_basic_line(prog_addr, "15 INPUT R");
235     prog_addr = save_basic_line(prog_addr, "16 K=1.6");
236     prog_addr = save_basic_line(prog_addr, "20 R2=R^2");
237     prog_addr = save_basic_line(prog_addr, "21 PRINT 'AREA =';3.14159*R2;' CIRCUM =';3.14159*2*R");
238     prog_addr = save_basic_line(prog_addr, "22 PRINT");
239     prog_addr = save_basic_line(prog_addr, "23 PRINT");
240     prog_addr = save_basic_line(prog_addr, "25 K1=K*R");
241     prog_addr = save_basic_line(prog_addr, "30 FOR X=-R TO R+0.1");
242     prog_addr = save_basic_line(prog_addr, "40 Y=K*SQR(R2-X^2)");
243     prog_addr = save_basic_line(prog_addr, "50 PRINT TAB(2.5+K1-Y);'*';TAB(5.5+K1+Y);'*'");
244     prog_addr = save_basic_line(prog_addr, "60 NEXT X");
245     prog_addr = save_basic_line(prog_addr, "70 END");
246 #endif
247

```

# SCELBAL

```
Untitled_0
New Open Save Connect Disconnect Options Clear Data View Help
Teensy: 4.1
Debug: 0
-----
SRAM1 Size: 65536 Bytes
SRAM1_START: 0x0
SRAM1_END: 0xFFFF
-----
=====
; SCELBAL by Mark G. Arnold (MGA) and Nat Wadsworth
; Copyright 1975 Scelbi Computer Consulting, Inc.
; All rights reserved
;
; SCELBAL is the only open-source, floating-point
; high-level language ever implemented on Intel's first
; general-purpose microprocessor, the 8008. It was
; published in book form:
; SCELBAL: A Higher-Level Language for 8008/8080 Systems
;
; ...This code originates from a version made by
; Steve Loboyko in 2001.
; This version has all 3 patches for SCELBAL.
;
; Big thanks to MGA for Retroschild use. 2025/02/24.
=====
=> type LIST to see the example drawing program.
.
LIST - see example circle program
RUN - run the example circle program
SCR - SCRatch. Start new program.
RESET Sequence - Initiate.
.
READY
Already loaded into memory.
LIST
10 PRINT 'RADIUS';
15 INPUT R
16 K=1.6
20 R2=R^2
21 PRINT 'AREA =';3.14159*R2;' CIRCUM =';3.14159*2*R
22 PRINT
23 PRINT
25 K1=K*R
30 FOR X=-R TO R+0.1
40 Y=K*SQR(R2-X^2)
50 PRINT TAB(2.5+K1-Y);'*';TAB(5.5+K1+Y);'*'
60 NEXT X
70 END
.
READY
```

```
Untitled_0
New Open Save Connect Disconnect Options Clear Data View Help
SCR - SCRatch. Start new program.
RESET Sequence - Initiate.
.
READY
LIST
10 PRINT 'RADIUS';
15 INPUT R
16 K=1.6
20 R2=R^2
21 PRINT 'AREA =';3.14159*R2;' CIRCUM =';3.14159*2*R
22 PRINT
23 PRINT
25 K1=K*R
30 FOR X=-R TO R+0.1
40 Y=K*SQR(R2-X^2)
50 PRINT TAB(2.5+K1-Y);'*';TAB(5.5+K1+Y);'*'
60 NEXT X
70 END
.
READY
RUN
RADIUS?10
AREA = 314.1592 CIRCUM = 62.83185
.
READY
```



- Play w/ a wide range of processors and software from 50yrs ago.
- Design your “hardware” and hack it till cows come home.
- Invent your own “peripherals” with zero cost \$ (DMA, UART, I/O)
- Play w/ cpu bus interface timings to see where cpu breaks.
  - Print low level bus activity cycle by cycle.
  - Be creative and squeeze out every clock for processor speed.

# Special Thanks to Mark G. Arnold (MGA)

- We have a rich computer history. As time goes on, naturally, companies shutdown, people leave, devices and files get fewer and even get lost.
- We can not steal their hard work just because they are not around.
- Companies and authors open-sourcing their commercial products is valuable. It enables new generation to experience and appreciate their products.
- I want to give special thanks to Mark G. Arnold (MGA) and the late Nat Wadsworth for enabling me to port SCELBAL to Retroshield 8008.
- I'm sure they weren't expecting SCELBAL to "gain" marketshare after 50 yrs.

# What's next for me?

- Every year I tell myself I will be done...
- 2025: Working on next batch of processors
  - **Transputer T805, T425, T225.**
  - Intel **8080** (to complete the Intel series)
  - Texas Instruments **TMS9995**
  - **Holding pattern:** NS32016, 8X305, DEC T-11, Fairchild F8, Motorola 6800

