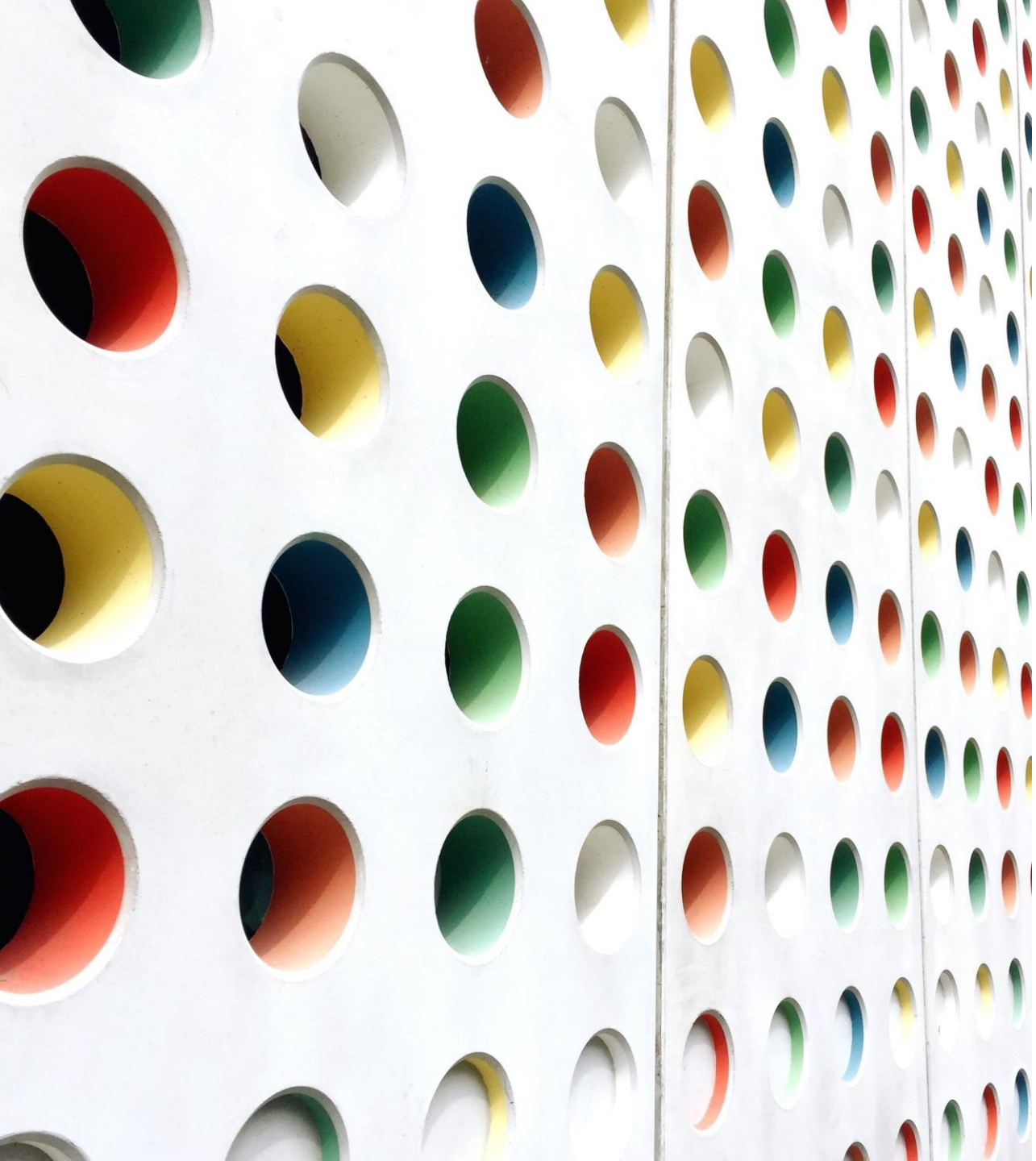


MicroCore Labs Emulators

HOW THEY WORK

HOW TO BUILD ONE

- CRAWFORD GRIFFITH
- VCF FORUM ID (CRAWFORD)

What We'll Cover

- Who I am, Who Ted Fried is, What MicroCore Labs is
- What the plug-in emulators are / aren't
- Detailed design review of Hardware and Software
- Build information, and gotchas
- Success story for debugging a vintage system
- Ideas for what you can do with these things
- More info – Links, contact info, stuff
- Thanks

Who I am, Who Ted Fried is, What MicroCore Labs is

- Me – VCF Member, into DEC, OSI, Making things
- Ted Fried – Embedded hardware engineer and vintage computer enthusiast with and interest in microprocessors
- MicroCore Labs - Blog - http://www.microcorelabs.com/home.html

  Github - https://github.com/MicroCoreLabs/Projects

  VCF Forum ID – MicroCoreLabs

  Youtube - https://www.youtube.com/@microcorelabs7698/featured

→ Ask questions, this is a talk not a lecture!←

Who I am, Who Ted Fried is, What MicroCore Labs is

MicroCore Labs is the collection of Ted Fried's websites

- Ted builds plug-in emulators, both FPGA* and microprocessor based
- All Open Source, All design and code published
- This talk - only about the microprocessor-based ones

* Field-Programmable Gate Arrays

Plug-in emulators - definitions

Definitions

"Emulators" -

Pure software emulators - All software running on something fast

Hardware based - Hardware or Hardware + Software inside a host (vintage) computer

"Cycle-accurate"

Virtually identical operation and speed as the original CPU, but not “cycle-exact”

"Plug-in emulators"

Replaces a CPU inside a system

Also known as In Circuit Emulator (ICE) - make Vanilla joke here

MCL microprocessor-based plug-in emulators

MicroCore Labs Plug-In Cycle-Accurate Emulators

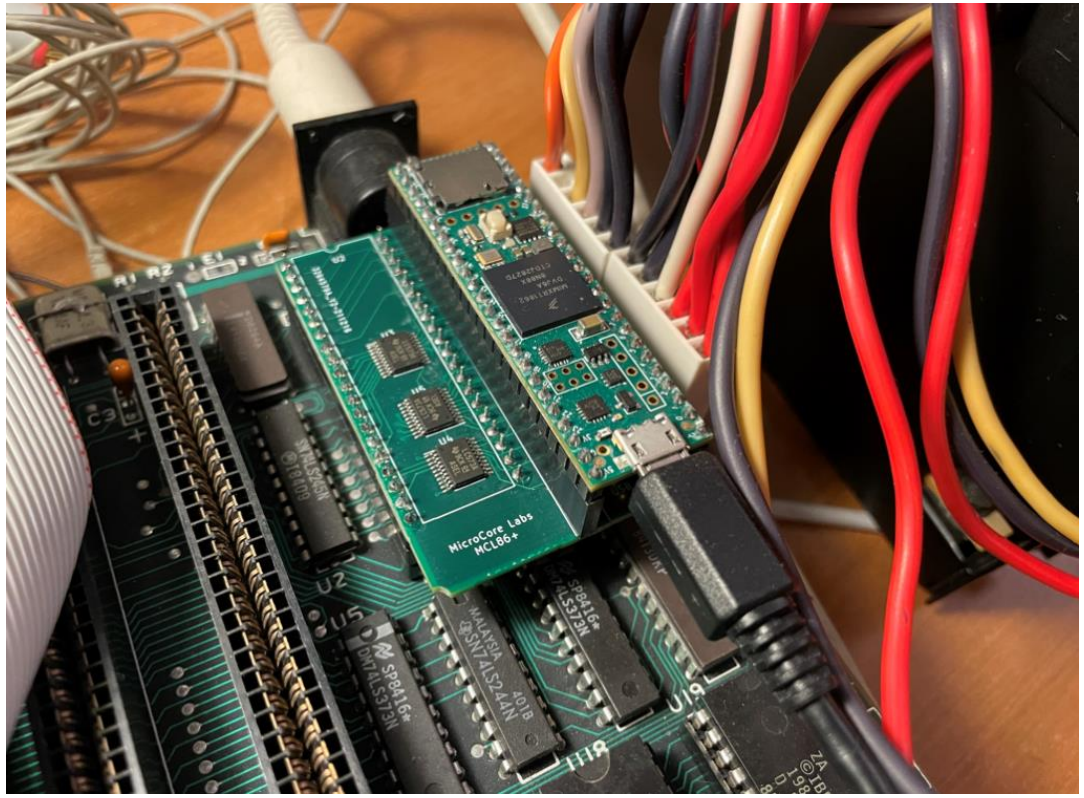
Naming "MCLxx" and "MCLxx+" - In General, the "+" ones are microprocessor-based

List of MCLxx[+] emulators

MCL65+	- 6502 *
MCL64	- 6510 (Commodore)
MCLZ8	- Z80 *
MCL86+	- 8088 *
MCL68+	- 68000

* I have personally built and tested these

A look at an MCL plug-in emulator



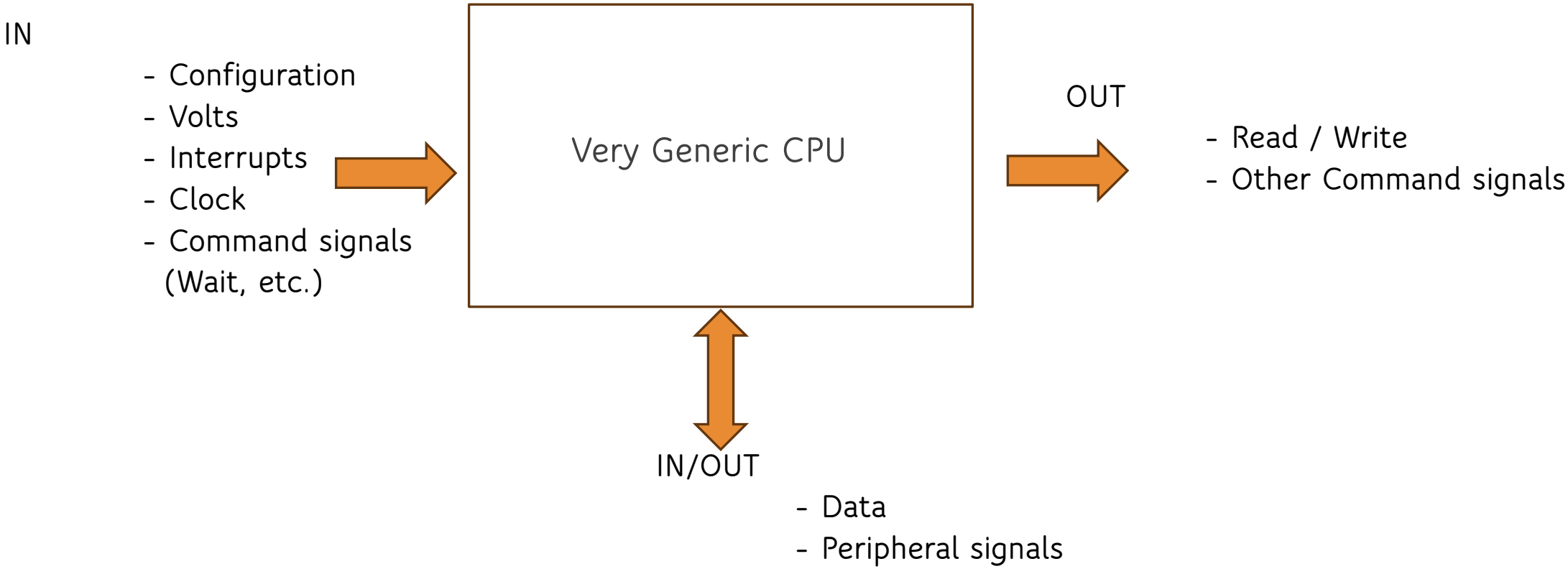
Here's what one looks like in a vintage machine (MCL86+ in an IBM PC 5150)

What the plug-in emulators can do

Why are these things useful?

- Can help debug a machine
- Can help bring up a machine w/o some things (boot ROM, disk, serial, I/O, video)
- Can emulate part of the system
- But, also to have fun! Do wild stuff like acceleration, Frankenstein brain swaps, etc.
- Extend/upgrade a vintage machine (discussed later)

Detailed design review of Hardware and Software – the design target



Detailed design review of Hardware and Software – Design considerations

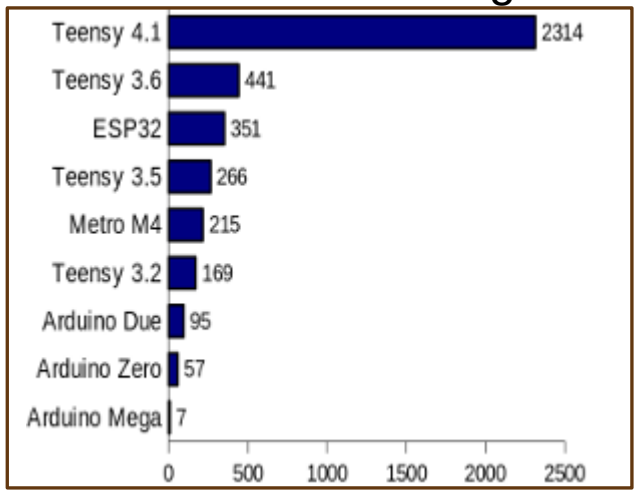
Considerations:

- Need something faster than the original CPU – Generally WAY faster
- Need something with a lot of memory (RAM, Flash)
- Need enough Input/Output lines to drive the CPU signals

Detailed design review of Hardware and Software

PJRC's Teensy 4.1

- ARM Cortex-M7 at 600 MHz
- Float point math unit, 64 & 32 bits
- 7936K Flash, 1024K RAM (512K tightly coupled), 4K EEPROM (emulated)
- QSPI memory expansion, locations for 2 extra RAM or Flash chips
- USB device 480 Mbit/sec & USB host 480 Mbit/sec
- 55 digital input/output pins, 35 PWM output pins
- 18 analog input pins
- 8 serial, 3 SPI, 3 I2C ports
- 2 I2S/TDM and 1 S/PDIF digital audio port
- 3 CAN Bus (1 with CAN FD)
- 1 SDIO (4 bit) native SD Card port
- Ethernet 10/100 Mbit with DP83825 PHY
- 32 general purpose DMA channels
- Cryptographic Acceleration & Random Number Generator
- RTC for date/time
- Programmable FlexIO
- Pixel Processing Pipeline
- Peripheral cross triggering
- Power On/Off management



CoreMark Benchmark

Detailed design review of Hardware and Software

Teensy 4.1

Teensy 4.1 Summary:

- Really fast CPU, on-board overclocking up to 816 Mhz without cooling

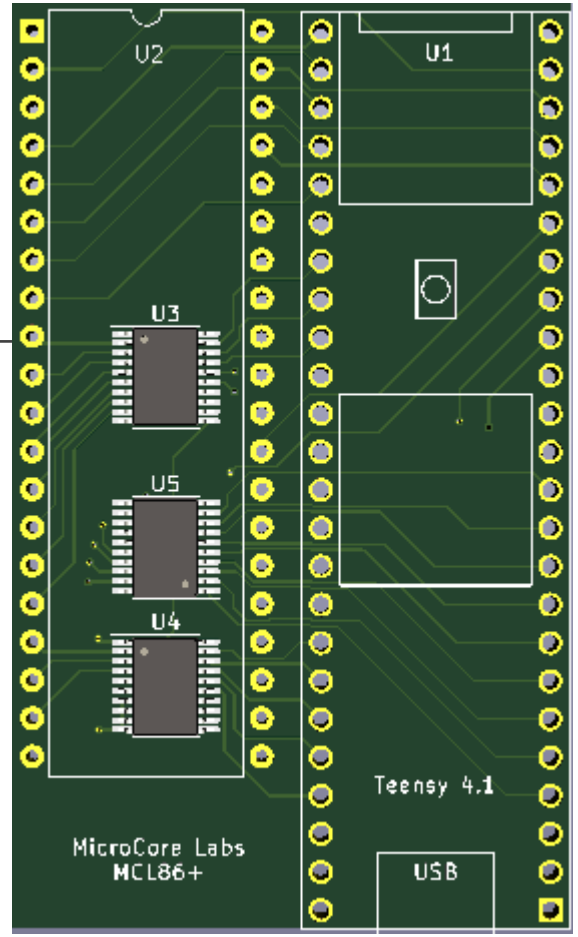
- Lots (55 pins) of Input/Output lines (42 are used on most MCLxx[+] boards)

- Software (a biggie) - ARDUINO (C/C++)
  LOL - Lots Of Libraries

- Cost is ~ $30 online and at MicroCenter stores

...but... Teensy is not 5V tolerant, it is a 3.3V board! (solved)

Detailed design review of Hardware and Software – PCB overview



Hardware – MCLxx[+] Boards

One Job – safely connect the Teensy 4.1 pins to the CPU Socket

Actual Boards (show some)

Fairly simple, 3-5 IC's, headers

buffers, latches mainly for 3.3V transition in and out of Teensy 4.1

Teensy on-board regulator takes in 5V and outputs 3.3V

Surface Mount Technology (scary!) - not really (discussed in the build section)

Detailed design review of Hardware and Software – Software major sections

Talk about that software – Relax, It's all in one Arduino code file (MCLxx.INO)

Parts of the emulator code:

- Setup - what Pins do what - In or Out
- Main Loop
  - get an instruction
  - decode it, execute it
  - rinse and repeat (Lots of waiting for the next clock cycle)
- Instruction Decode/Execute functions
  - as many as it takes
- Bunch of Arrays
  - Some arrays are for ROM images
  - Some arrays are for Pin value lookups (discussed next slide)
- Bus Interface
  - Memory and I/O
  - Arbitration for what goes where host system vs Teensy

Detailed design review of Hardware and Software – Software special features

Software – Secret Sauce for Speed

Teensy-specific Code

Fast I/O

- Standard Arduino library routines for GPIO* read and write are too slow
- Teensy allows direct access to GPIO registers
- "without the fast IO none of this would be possible." – Ted Fried

Pin value lookup Arrays

- Pin values sent to GPIO registers (and then to CPU socket)
- Physical Pins are not always neatly laid out on target CPU or Teensy
- Either have to do bit manipulation (logical OR, shifts) or lookup arrays
- Array lookups are faster than inline math
- Only if you change a MCLxx[+] board layout should this concern you

* General Purpose Input Output

Detailed design review of Hardware and Software – Software setup overview

Software Realities

- Have to install the Arduino IDE on a PC or Mac
- Have to install Teensyduino utility, from PJRC
- Have to set the Processor Speed in Arduino IDE menus*
- Have to set the Code Optimization level (hint – use Fastest)*

* Or use my forked code in github

Build information – Obtaining PC Boards

MCLxx[+] Boards - Got to make some boards! (or more accurately, get them made - best to farm these out)

Overseas manufacturing (pluses and minuses)

JLCPCB and PCBWay are two companies that do this

Price-per-board is low ($25 or less for 5 boards)

Made very quickly (usually 48 hours or less)

Shipping is high (like $25) (or wait a few weeks)

You can get a solder paste stencil (if you're fancy)

Both JLCPCB and PCBWay can also assemble the SMT for you,

JLCPCB with their parts, and PCBWay with parts from Digikey, Mouser, etc.

Best to do a group buy, spread the cost

Build information – Obtaining parts

Build Section – Board Components

Bare Boards:

Get the IC's from anywhere (Mouser, Digikey, eBay)

They are TSSOP-20 (0.65mm pin spacing) exception: DIP packages for MCL65+

Remember to get 3.3V chips (5V tolerant), generally 74HC or 74LVC or 74AHC

All Boards:

Headers - get 40 pin Male-male machine pin for the plug into the CPU socket (PCB bottom)

- get 40 pin male and female "Arduino-like but longer" (PCB top)

- Teensy 4.1 is 48 pins (24 to the side)

Build information – Soldering IC's (PCB vendor option)

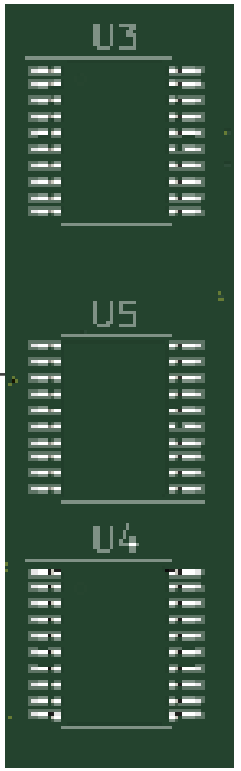
Build Section – As Easy as 1-2-3! - Soldering 1a

---------- OPTION A ----------

Have the PCB shop assemble the SMT

Most have this service

Build information – Hand soldering Option



Build Section – Soldering 1b – it's not terribly hard

---------- OPTION B ----------

Good soldering station, fine soldering tip, thin solder, solder wick and a MAGNIFYING GLASS

Surface Mount Technology (SMT) chips - YOICKS!

LOOK AT THE LAYOUT-HARD TO SEE THE PIN 1 INDICATION, THE CHIPS DO NOT ALL HAVE THE SAME ORIENTATION

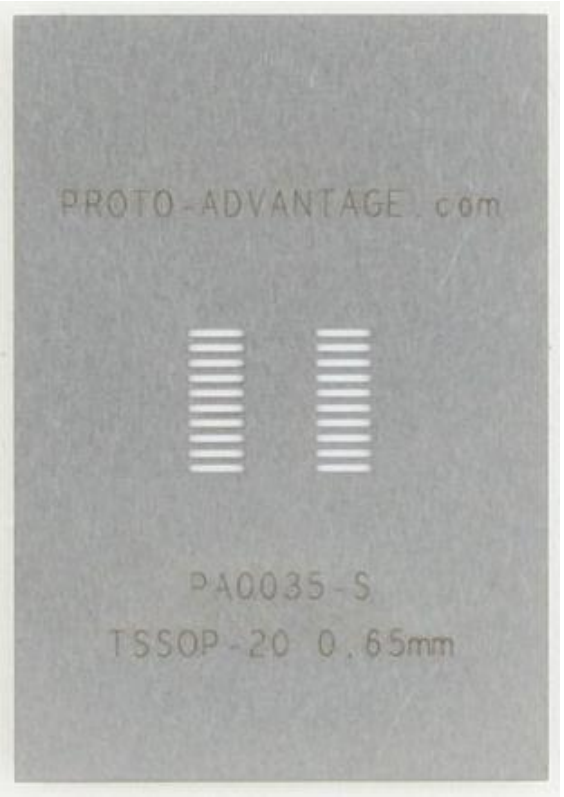
Steady hands - these are TSSOP-20 0.65mm chips

Center the chips, solder pin 1, recenter and solder pin 11 (opposite corner)

LOOK AT THE PIN ONE INDICATORS, IT IS BETTER TO DESOLDER 2 PINS THAN 20! DO NOT ASK ME HOW I KNOW THIS!

Solder the rest of the pins

Build information – Solder stencil, paste, hot air option



Build Section – Soldering 1c

---------- OPTION C ----------

Use the solder stencil

Apply solder paste with a credit card

Peel off the solder stencil carefully

Center the chips carefully on the solder

Pre-heat board on a hot plate

Hit it with the hot air station (heat high, air low so you don't send chips airborne)

Watch with wonder as the chips magically move to the exact right place

Build information – Soldering Teeny headers

Build Section – Soldering 2 – Teensy Headers

- Inspect the SMT chip pins for bridges, use solder wick dipped in flux to fix
- Install Teensy headers
  - Solder male "Arduino-like" headers to the underside of the Teensy 4.1 board (if not supplied)
  - Long "Arduino-like" female headers on top of the PCB, use the Teensy w/headers to hold them parallel
  - Push the header up from the underside, tack down the end pins
  - Check that the headers are tight to the board
  - Solder the other pins down

Build information – Soldering CPU headers

Build Section – Soldering 3 – CPU Headers

Install CPU headers

put the side of the header with thinner pins into a 40 pin socket

put the headers through the underside of the PCB

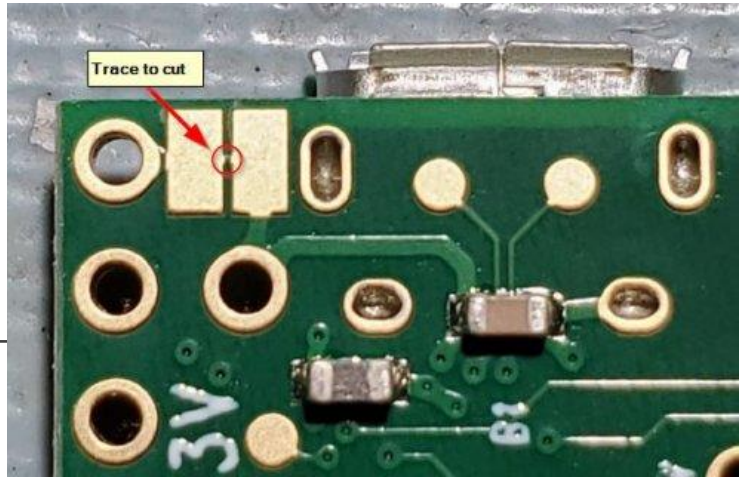
turn the board over holding the socket+headers tight to the underside of the board

solder the end pins (1, 20, 21, 40)

check the headers are tight to the board

solder the other pins down

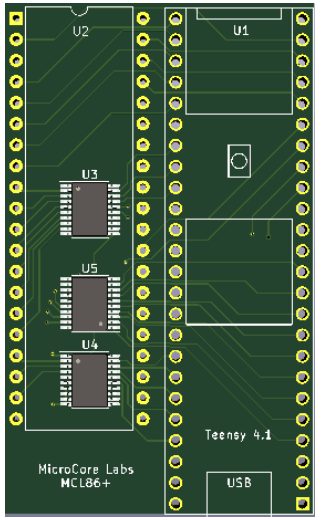
Build information - Final assembly



Build Section - final assembly

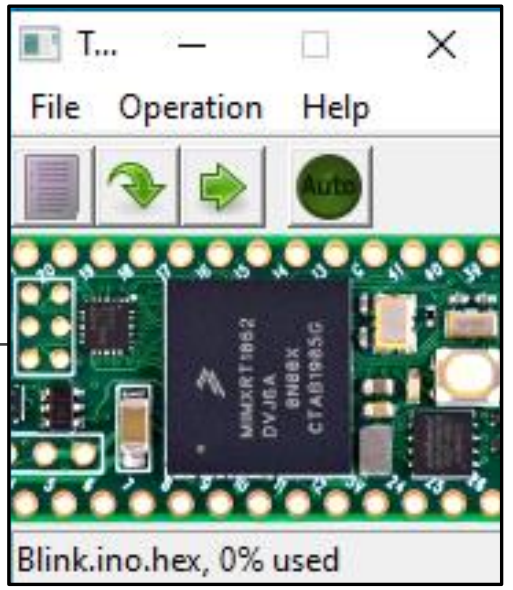
- Give the Teensy 4.1 some minor surgery (or delay this until you program the Teensy)
- Cut the pad near Vcc, so that the USB cable does not try to power the target system!
- Plug the Teensy 4.1 board into the header

  (look at the outline and make sure the sd card holder and usb are lined up)



Build information – Compile and flash software

Build section - Software



- Download the software for the board from Github (RAW Download)
- Install the Arduino IDE or update yours to the latest, install Teensyduino from PJRC.com
- Launch the Arduino IDE
- Go to board manager and search for Teensy, load the library
- Start a new project and navigate to the MCLxx.INO file and open it
- Immediately set the board to Teensy 4.1
- Set the Speed to 816Mhz
- Set the Optimization to Fastest
- Cross fingers
- Plug in MCLxx[+] board into a power source if the Vcc pad was cut
- Plug a USB micro cable into the Teensy (make sure it is not a charge-only cable)
- Cross fingers and hit the compile/Verify button on the IDE
- When the Teensy programmer pop-up shows, press the white button on the Teensy
- Smile when the programmer reports the teensy is programmed and reset okay

Success Story for debugging a vintage machine

- I have a Ohio Scientific Challenger 1P, with no serial port and a malfunctioning floppy disk drive
- I had built a MCL65+ board, and tested it on a (loaned) Apple II+
- David Gesswein has an excellent floppy disk program for OSI computers (**Floppy test 1.04u.zip**)
  - And I had no way to load it other than a somewhat flaky keyboard on the OSI C1P

So…

- I converted David's Floppy Test code in Hex to a C-code array using Notepad++
- I pasted the array into the MCL65.INO file
- I made a small patch in the MCL65+ code to load the array at address 0x0300
- I plugged the MCL65+ into the Challenger 1P, hit reset, entered the monitor, hit 0300 G
- And *BAM* there was the disk test program running on my C1P!

(Total elapsed time was ~1 hour)

Success Story for debugging a vintage machine

```
//CG - Add
// David Gesswein's OSI Disk Test in Hex, load to 0x300 in RAM
uint8_t Disk_Test[0x12EB] = {
0x78,0x20,0x11,0x13,0xAD,0x01,0xFE,0xF0,0x10,0xA9,0x00,0xAA,0x8D,0x00,0xDF,0x2C,0x00,0xDF,0x30,0x02,0xE8,0x2C,0xA9,0x80,0x2C,0xA9,0x40,0xCA,0x85,0x30,0x86,0x31,
0xA9,0x15,0x85,0x2C,0xA2,0x0A,0x20,0x0B,0x10,0xA2,0x00,0x20,0xDD,0x12,0xB0,0x01,0xCA,0x20,0xA4,0x12,0x86,0x32,0x20,0x1A,0x10,0x0D,0x0A,0x48,0x49,0x54,0x20,0x41,
0x4E,0x59,0x20,0x4B,0x45,0x59,0x20,0x54,0x4F,0x20,0x53,0x45,0x4C,0x45,0x43,0x54,0x20,0x43,0x4F,0x4E,0x53,0x4F,0x4C,0x45,0x20,0x44,0x45,0x56,0x49,0x43,0x45,0x0D,
0x0A,0x00,0x20,0xA4,0x12,0xCA,0xE0,0xFE,0xD0,0xC7,0x20,0xA4,0x12,0xB0,0x0C,0x20,0x73,0x12,0x90,0xF6,0xA2,0xFF,0xA0,0x04,0x4C,0x7F,0x03,0xA2,0x00,0xA0,0x06,0x86,
0x32,0x84,0x73,0xA9,0x00,0x85,0x2D,0x85,0x33,0x85,0x63,0x85,0x36,0x85,0x2E,0xA9,0x01,0x85,0x35,0xA9,0x18,0x85,0x34,0x24,0x30,0x50,0x03,0x4C,0x6C,0x04,0x4C,0x59,
0x04,0x20,0xA4,0x10,0x20,0xA4,0x12,0xA2,0x00,0x20,0xBC,0x12,0x20,0xED,0x12,0xC9,0x31,0xD0,0x03,0x4C,0xF4,0x08,0xC9,0x32,0xD0,0x03,0x4C,0x92,0x04,0xC9,0x33,0xD0,
0x03,0x4C,0xF5,0x05,0xC9,0x34,0xD0,0x03,0x4C,0xF7,0x03,0xC9,0x35,0xD0,0x03,0x4C,0xE9,0x04,0xC9,0x36,0xD0,0x09,0xA9,0xFF,0x45,0x36,0x85,0x36,0x4C,0xA1,0x03,0xC9,
0x37,0xD0,0x03,0x4C,0x3B,0x06,0xC9,0x38,0xD0,0x03,0x4C,0x82,0x14,0xC9,0x39,0xD0,0x03,0x6C,0xFC,0xFF,0x4C,0x51,0x04,0x20,0x1A,0x10,0x0D,0x0A,0x45,0x6E,0x74,0x65,
0x72,0x20,0x79,0x6F,0x75,0x72,0x20,0x64,0x69,0x73,0x6B,0x20,0x64,0x72,0x69,0x76,0x65,0x20,0x74,0x79,0x70,0x65,0x20,0x28,0x38,0x29,0x20,0x69,0x6E,0x63,0x68,0x2C,
0x20,0x28,0x35,0x29,0x2E,0x32,0x35,0x20,0x69,0x6E,0x63,0x68,0x2C,0x20,0x6F,0x72,0x20,0x28,0x33,0x29,0x2E,0x35,0x20,0x69,0x6E,0x63,0x68,0x3F,0x20,0x3E,0x00,0x20,
0xBC,0x12,0x20,0xED,0x12,0xC9,0x38,0xF0,0x23,0xC9,0x35,0xF0,0x0C,0xC9,0x33,0xF0,0x2E,0xA9,0x3F,0x20,0xED,0x12,0x4C,0xA1,0x03,0xA9,0x28,0x85,0x2A,0xA9,0x0A,0x85,
0x2B,0xA9,0x89,0x85,0x5E,0xA9,0xF7,0x85,0x5F,0x4C,0xA1,0x03,0xA9,0x4D,0x85,0x2A,0xA9,0x0F,0x85,0x2B,0xA9,0xF0,0x85,0x5E,0xA9,0xF1,0x85,0x5F,0x4C,0xA1,0x03,0xA9,
0x50,0x85,0x2A,0xA9,0x10,0x85,0x2B,0xA9,0x34,0x85,0x5E,0xA9,0xEF,0x85,0x5F,0x4C,0xA1,0x03,0x20,0x1A,0x10,0x0D,0x0A,0x45,0x6E,0x74,0x65,0x72,0x20,0x32,0x20,0x64,
0x69,0x67,0x69,0x74,0x20,0x64,0x65,0x63,0x69,0x6D,0x61,0x6C,0x20,0x74,0x72,0x61,0x63,0x6B,0x20,0x74,0x6F,0x20,0x74,0x65,0x73,0x74,0x20,0x3E,0x20,0x00,0x20,0xCF,
0x05,0x90,0xCF,0x85,0x62,0xC5,0x2A,0x10,0xC9,0x20,0x32,0x0F,0x20,0x57,0x0F,0x20,0x74,0x0F,0x90,0x03,0x4C,0xFB,0x0E,0xA5,0x62,0xF0,0x07,0x20,0xD8,0x0F,0xC6,0x62,
0xD0,0xF9,0xA9,0x01,0x85,0x74,0x4C,0xF8,0x08,0xA9,0x00,0x85,0x33,0x20,0x1A,0x10,0x0D,0x0A,0x45,0x6E,0x74,0x65,0x72,0x20,0x68,0x65,0x78,0x20,0x66,0x69,0x6C,0x6C,
0x20,0x70,0x61,0x74,0x74,0x65,0x72,0x6E,0x20,0x6F,0x72,0x20,0x73,0x70,0x61,0x63,0x65,0x20,0x66,0x6F,0x72,0x20,0x72,0x61,0x6E,0x64,0x6F,0x6D,0x20,0x3E,0x20,0x00,
0x20,0xBC,0x12,0x20,0xED,0x12,0x29,0x7F,0xC9,0x20,0xF0,0x7D,0x20,0xAE,0x05,0x90,0xB8,0x0A,0x0A,0x0A,0x0A,0xAA,0x20,0xBC,0x12,0x20,0xED,0x12,0x20,0xAE,0x05,0x90,
0xA8,0x85,0x2F,0x8A,0x05,0x2F,0x85,0x34,0x20,0x1A,0x10,0x0D,0x0A,0x45,0x6E,0x74,0x65,0x72,0x20,0x74,0x77,0x6F,0x20,0x64,0x69,0x67,0x69,0x74,0x20,0x70,0x61,0x73,
```

The Code Array – Covers several screens

Success Story for debugging a vintage machine

```
inline uint8_t internal_address_check(int32_t local_address) {
  //CG Add
  if ( (local_address>=0x0300) && (local_address <0x15EB)) return 0x01 ; //use the ROM image in 0x0300-0x15EB
  //End CG Add
  if ( mode == 0) return 0x0; // Use only external memory (on target system)
  if ( (local_address>=0x0000) && (local_address <0x0400))   return mode;   //  6502 ZeroPage and Stack
  if ( (local_address>=0x0400) && (local_address <0x0C00))   return 0x1;    //
@ -1775,7 +1934,10 @@ void opcode_0xAB() {  Fetch_Immediate();    Begin_Fetch_Next_Opcode(); return;
//
// ------------------------------------------------
 void loop() {

//CG Add
  uint16_t i ; //Address counter
//END CG add

  //setup();

  // Give Teensy 4.1 a moment
@ -1786,7 +1948,11 @@ void opcode_0xAB() {  Fetch_Immediate();    Begin_Fetch_Next_Opcode(); return;


  reset_sequence();

//CG Add - fill up local RAM area
 for (i=0x0300;i<0x15EB;i++) {
  internal_RAM[i] = Disk_Test[i-0x300];  //
 };
```

The Logic – Tell the emulator to use internal RAM for 0x300-0x15EB, define a counter, put the array in internal RAM

Ideas for what you can do with these things

Accelerate your machine – Ted has already added this in most of the existing code

Write code [or port] a system test process in C and run it on the Teensy
-> there is already one done for the Commodore 64 !

Get a machine to boot that does not have some vital hardware, like tape (paper or plastic),
serial port, video out, keyboard, diskette or Hard disk

Easily mirror, or even increase system RAM size or try different ROM versions using emulated memory inside the Teensy

Expand a machine using the SD card or fast serial over USB - teensy as a USB host

Add networking with Ethernet (yeah, Teensy 4.1 has that too)

Emulate an extremely rare piece of hardware (like the OSI Hard disk)

Put a new (or expanded) instruction set into your computer (see the MicroCore Labs 68000 in an IBM PC)

Emulate another CPU : 65816, 6800, 6809, PDP-11, Harris 6120 ...

S U C C E S S !


PCBWay
SPARKPOST
OS-65D Tutorial disk five - Sept. 16, 1981
1 > Directory
2 > Create a new file
3 > Change a file name
4 > Delete file from diskette
5 > Create blank data diskette
6 > Create data diskette with files
7 > Create buffer space for data files
8 > Single or dual disk drive copier
9 > Enter OS-65D system
OHIO SCIENTIFIC
MINI FLOPPY DISKETTE
BUSINESS DISK 2
WARNING!
MINI FLOPPY DISKETTE
MICROCOMPUTER SOFTWARE
SCIENTIFIC
OHIO

More info – Links, contact info, stuff

Crawford's forks of MCL projects - Github.com/CrawfordGriffith

github.com/CrawfordGriffith/MCLProjects-MCL65plus (for MCL65+ 6502)

github.com/CrawfordGriffith/MCLProjects-MCL86plus (for MCL86+ 8088)

github.com/CrawfordGriffith/MCLProjects-MCLZ8 (for MCLZ8 Z80)

Amazon Link for Male-male machine pin headers (for CPU socket)

Amazon Link for 40-pin "Arduino-like" heaters (male and female)

PJRC Downloads – for Teensyduino utility

Thanks !!!!

You folks for coming to the talk!

Ted Fried for his AWESOME projects and help and feedback!

Jeff Brace and volunteers for putting on an (always) AWESOME Vintage Computer Festival East event!

David Gesswein for his Floppy Test program!

Alex Jacocks and Maki Kato for loaning me systems to test on!