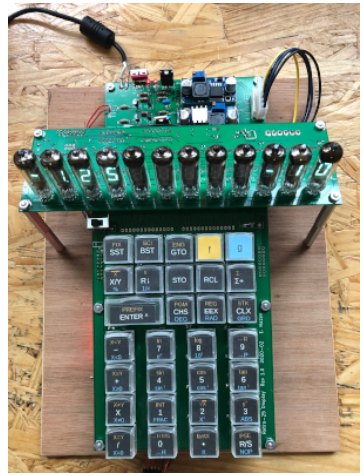
The Retro-25 DIY Calculators

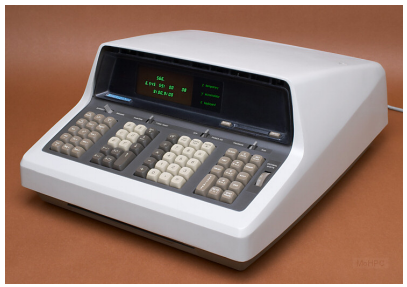



E. Hazen

October 8, 2020

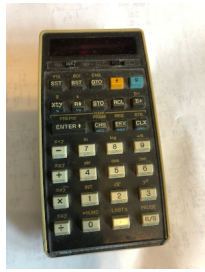
Personal and Professional History...

I've been fascinated by electronics since I was very young!



At age 13 (in 1973) I lived briefly in Leeds, England and worked at the Uni as an unpaid physics "graduate assistant". I encountered an HP-9100 and taught myself to program it. I was immediately hooked!

In High School I made the acquaintance of an HP-2100 series computer running HP timeshare BASIC. We played various games (some legit, like TREK73) and others (posting the administrator password on the chalkboard) not so legit.



In 1975 my father bought an HP-25. I devoted myself to delivering the *Ann Arbor News* faithfully to about 60 subscribers, and eventually I could afford my own HP-25 at $195.00 (about £663 today)

Unfortunately both HP-25 are now dead; mine fell in the fish tank with the piranhas while I was away and my roomies were scared to retrieve it! My father's died when the batteries lost contact.

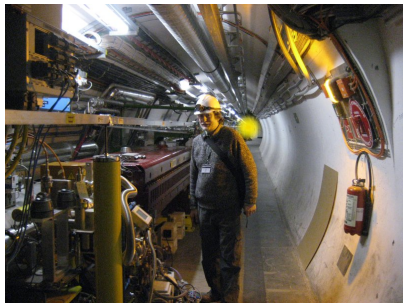
...more history

I studied computer science for a while at Michigan and became expert with the model 29 keypunch (I hated that thing!) I did learn a few other things (ALGOL and S/370 assembly)

What really changed my life was working at Radio Shack. They introduced the TRS-80 and I immediately bought one (well, honestly, I put it on "lay-a-way" at my house!) I taught myself Z80 assembly language and wrote a simple multi-tasking kernel and FORTH system.

I parlayed my Z80 skills into a job at ETC (theatre lighting). I designed their Idea^{TM} and Vision^{TM} lighting consoles, and wrote much of the low-level software in Z80 assembly (using my trusty TRS-80, now decked out with dual 8 inch floppies) as a dev system.

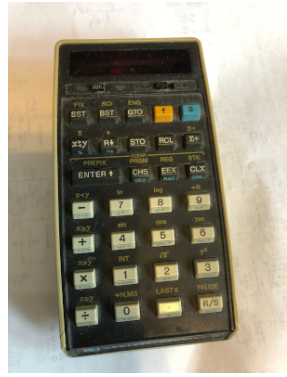
Since then I've worked at Boston University building electronics mostly for the LHC at CERN. I've also become somewhat of an HP collector, acquiring a 10C, 11C, 15C, 16C and 67. The 10C and 15C were filched, the 11C and 16C I use almost daily.



Why? Why? Why?

Why a DIY HP-25?

- The HP-25 is my favorite calculator
- The display and keyboard are too small for middle-aged eyes
- I hate mouse- and touchscreen-based emulators (no tactile feel)
- I have a bunch of time on my hands (*NOT!*)



Why use a Z80?

- I know them inside-out
- They were released about when the HP-25 came out
- I like designing and assembling good old thru-hole DIP boards

Why the HECK VFD displays?

- Because they are *cool!*
(and nixies are a lot of trouble)

Specification

Well, this is a hobby project so no spec needed, but I had goals:

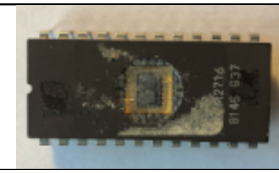
- Big, clicky buttons; display I can read without glasses
- Simple hardware I could solder easily
- Cross-development from my Linux machine

Parts! Parts! Parts!

I always start with the parts.

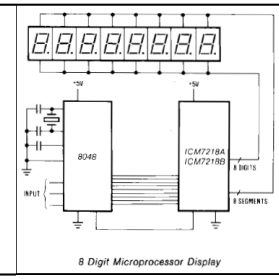
CPU: The Z80 is a no-brainer (initially Z80A, 4MHz)

Memory: first considered UV EPROM (nah!)
AT28C256-15 EEPROM and 62256LP-70 RAM (both 32k, DIP-28)



Display:
V1: 10mm red 10mm red common anode 7-segment
V2: IV-6 green vacuum fluorescent display tubes

Driver: ICM7218A MUX driver. I used these for theatre lighting consoles back in the day and they work well.

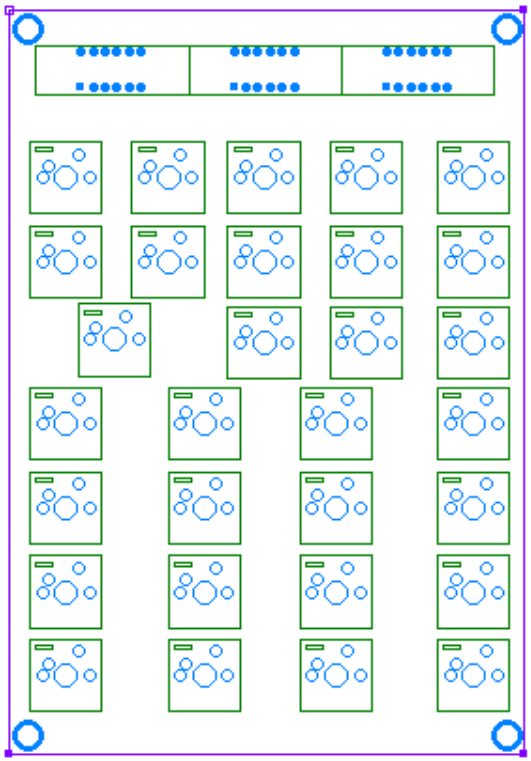


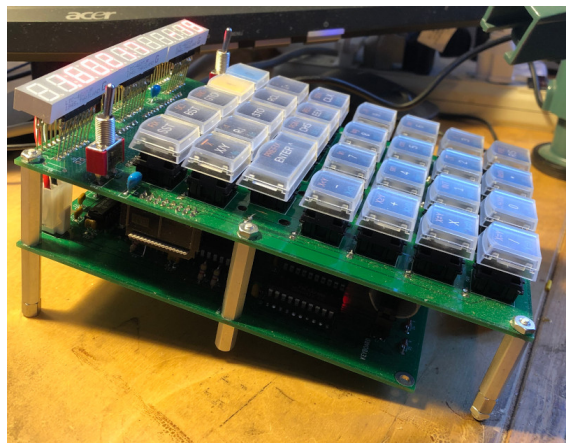
Switches: Cherry MX1A mechanical. These are my favorites!

Otherwise, I have drawers full of 74LS and 74S series TTL at work because we can't bear to throw anything....

Boards!

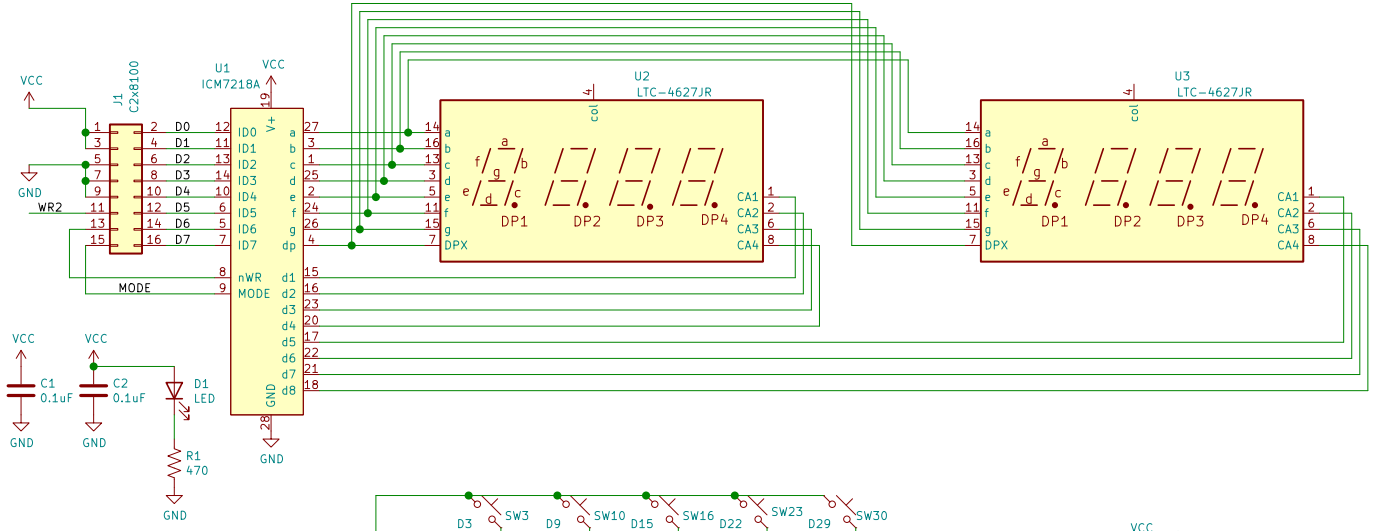
How many boards? Started with a "toy" layout in ExpressPCB to see how everything fit. Looked like about 120x170mm. So, I fired up KiCAD and made a schematic and did a layout for a Display/Keyboard.

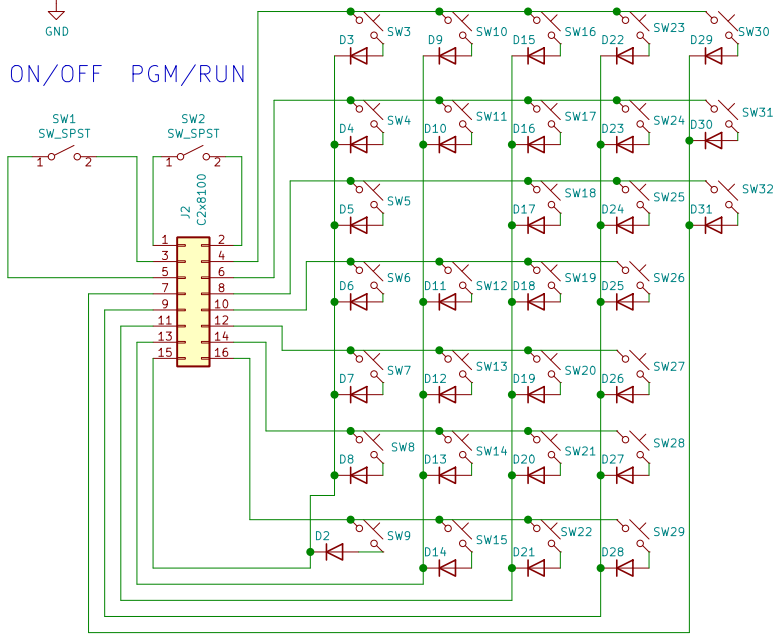


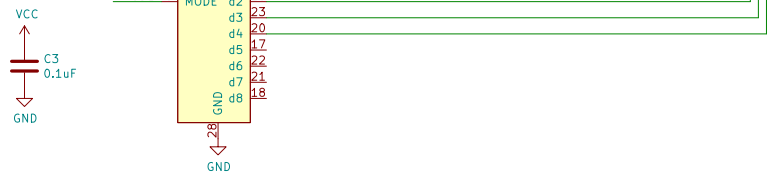


- Doesn't look like the CPU etc will fit
- Planning for a two-board stack

Schematics (KB, Display)





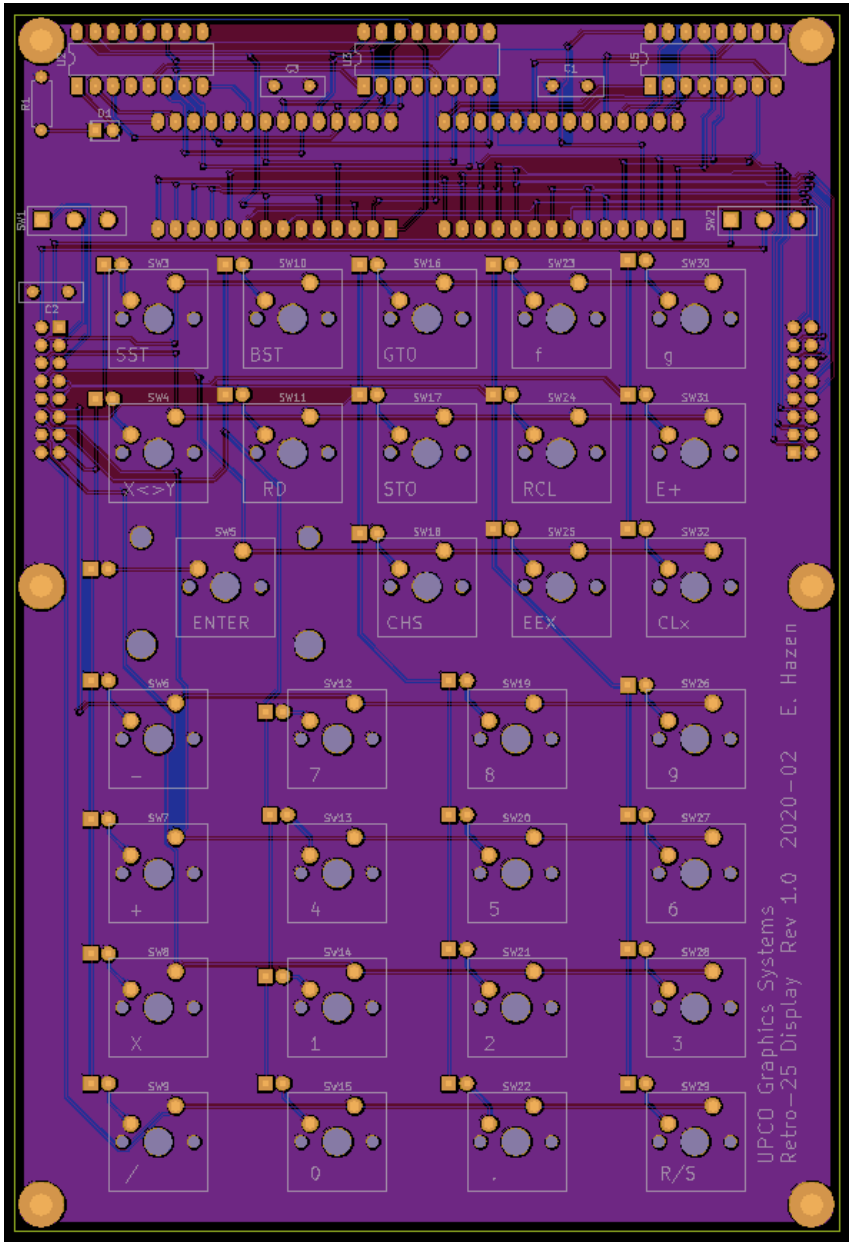


UPCO Graphics Systems
Retro-25 Display Panel

Keyboard is standard matrix

Display handled by two ICM7218A.
Almost no additional parts

PCB (KB, Display)



Placement and routing pretty simple.
Two header connectors, one for KB and one for LEDs

Top Cu area to GND
Bottom Cu area to VCC

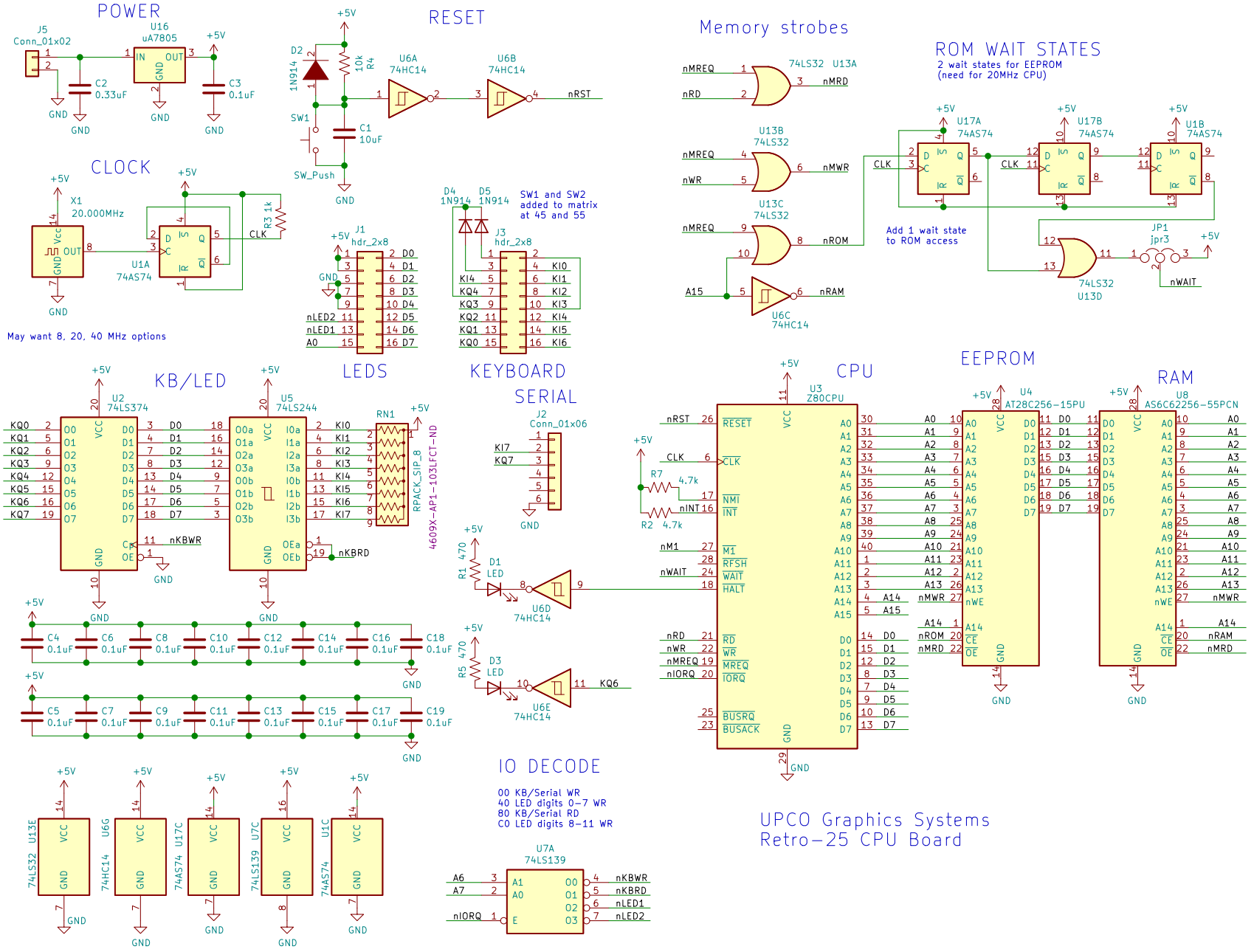
Design rules based on what JLCPCB could do easily:

- 6 mil line/space (0.15 mm)
- 20 mil via with 10 mil hole (0.5mm / 0.25mm)

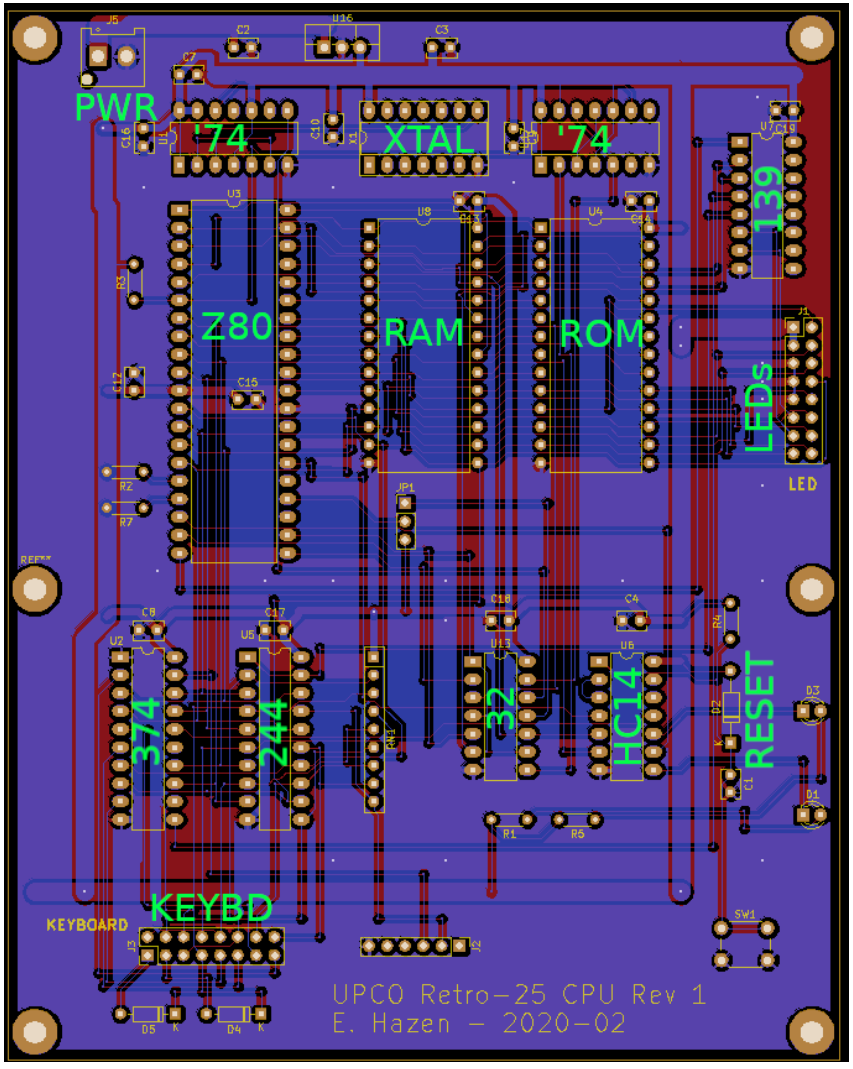
Switches and LEDs take up most of front

Diodes and LED drivers on back

Schematics (CPU)



PCB (CPU)

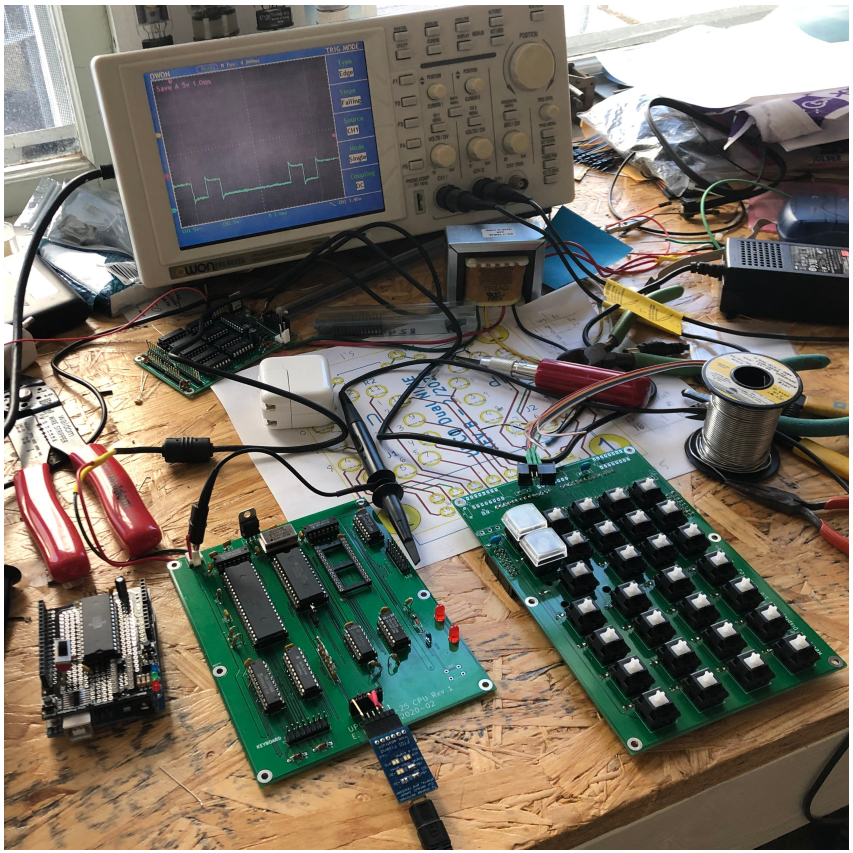


Placement and routing pretty simple.
Two header connectors, one for KB and one for LEDs
All components on the top.

Top and bottom Cu areas to GND

CPU, RAM, EEPROM in top row

Bringing it Up! (hardware)



The boards and parts are here!

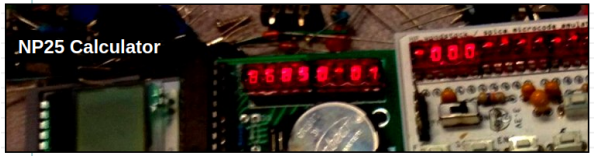
- Easy soldering (all thru-hole)
  But then what? Software?
- Brief diversion to make a junk-box EEPROM programmer
- First program: delay, then HLT
  i.e. `2B 7C B5 20 FB 76`
  (glad I put a HALT LED on the board!)
- Then a struggle with bit-bang serial (next time use a UART!)
- Then write a debug monitor... "umon2".

Two months later...

Bringing it Up! (software)

NONPAREIL
High-Fidelity Calculator Simulator

http://nonpareil.brouhaha.com



http://simpleavr.github.io/NP25/

Z88DK

https://github.com/z88dk/z88dk

```
#######  #####   ###          #####   ###  #    #
   #  #     # #  #  #        #    #   #   ##  ##
   #  #     # #     #        #        #   # # # #
   #   #####  #     #  #####  #####   #   #  #  #
   #  #     # #     #              #   #   #     #
   #  #     # #  #  #        #     #   #   #     #
   #  #####   ###          #####   ###  #    #
```

https://www.autometer.de/unix4fun/z80pack/

Download and study...

- Nonpareil by Eric Smith
  Very complete, but quite complex as it supports many models
- NP25 "Nonpareil Physical" by Chris Chung
  Woodstock-only emulator for MSP430
  Much more promising

In the end I took some of both, and got an HP-25 only simulation running in plain C under linux, using a tty for input and output.

Then, I built it using sdcc in Z88DK and tested it under the z80pack simulator.

Bringing it Up! (hardware+software)

Lots of loose ends...

First, write a serial boot loader

Then write a C program to send hex files

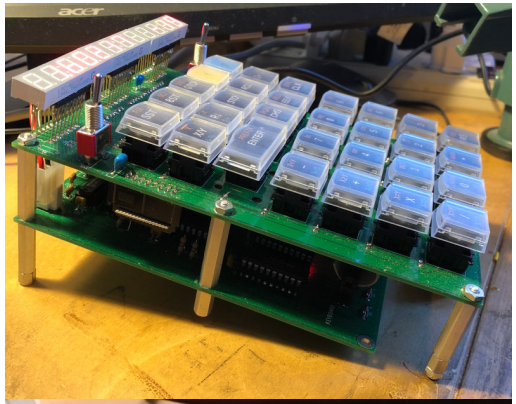
Then, work on the monitor...

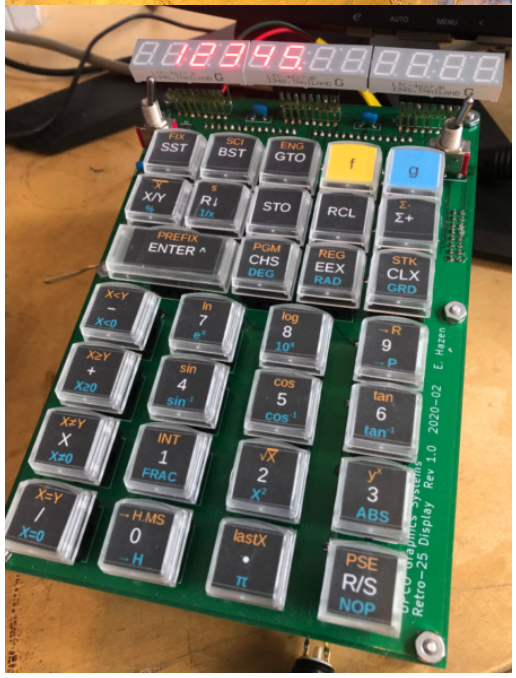
```
d <addr> <count>          dump memory
e <addr> <dd> <dd>...     edit up to 16 bytes in memory
o <port> <val>            output <val> to <port>
z <val>                   set port zero value bits 0-6
i <port>                  input from <port> and display
g <addr>                  goto addr
b <addr>                  set breakpoint (currently 3-byte call)
a <val1> <val2>           hex Arithmetic
c                         continue from breakpoint
c <addr>                  continue, set new breakpoint
m <start> <end> <size>    memory region compare
p <start> <end> <size>    memory region copy
l                         binary load from serial
r                         repeat last command
k                         scan HP keyboard
f <addr>                  dump HP registers from <addr> (A)
7 <addr>                  update 7-segment display from <addr>
V <addr>                  update VFD display from <addr>
```

I eventually needed all of those commands...

Finally, start to load and debug the calculator
Before long, it works!!

Wrapping it Up





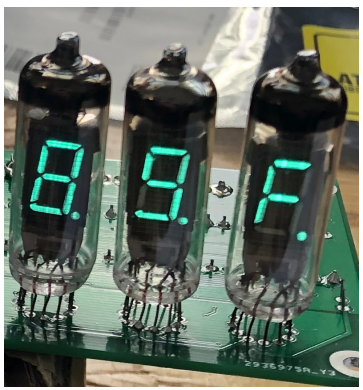
Stack the boards, add "on/off" and "prgm/run" switches, Make a simple wood box, it's done!

What Next?

S/N 2, of course!

LEDs are boring!
Nixies are troublesome!

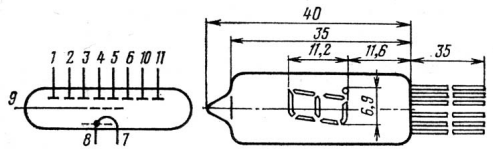
Let's try VFDs... never used them before



My Russian isn't so good, but Google Translate to the rescue...

ИВ-6

Индикатор вакуумный люминесцентный одноразрядный для отображения информации в виде цифр, букв, точки.
Оформление — стеклянное, сверхминиатюрное. Индикация производится через боковую поверхность баллона. Размер знакоместа 6,9×11,2 мм. Изображение формируется из светящихся анодов-сегментов. Цвет свечения — зеленый. Масса 11 г.
Выводы электродов: *1—6, 10, 11* — аноды-сегменты; *7* — катод, проводящий слой на внутренней поверхности баллона; *9* — сетка (маска).



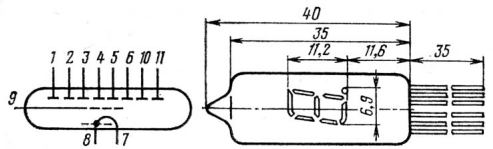
Основные данные

Яркость свечения	300—600 кд/м²
Угол обзора	⩾80°
Ток накала	50±5 мА
Ток анода-сегмента:	
в статическом режиме	⩽0,5 мА
в импульсном режиме	⩽2 мА
Ток анодов суммарный (постоянный)	⩽1,8 мА
Ток сетки:	
в статическом режиме	⩽10 мА
в импульсном режиме	⩽45 мА
Напряжение накала	$1{,}2^{+0{,}15}_{-0{,}2}$ В
Напряжение анодов и сетки:	
в статическом режиме	25—30 В
в импульсном режиме	50—70 В
Наработка	⩾10 000 ч

Примечание. Для получения цифр и букв рекомендуется подключать выводы анодов следующим образом: цифра 1 — *1, 10*; цифра 2 — *1, 2, 3, 5, 6*; цифра 3 — *1, 2, 3, 6, 10*; цифра 4 — *1, 3, 4, 10*; цифра 5 — *2, 3, 4, 6, 10*; цифра 6 — *2, 3, 4, 5, 6, 10*; цифра 7 — *1, 2, 10*; цифра 8 — *1, 2, 3, 4, 5, 6, 10*; цифра 9 — *1, 2, 3, 4, 6, 10*; цифра 0 — *1, 2, 4, 5, 6, 10*; точка — *11*, буква А — *1, 2, 3, 4, 5, 10*; буква Б — *2, 3, 4, 5, 6, 10*; буква Г — *2, 4, 5*; буква Е — *2, 3, 4, 5, 6*; буква З — *1, 2, 3, 6, 10*; буква Н — *1, 3, 4, 5, 10*; буква О — *1, 2, 4, 5, 6, 10*; буква П — *1, 2, 4, 5, 10*; буква Р — *1, 2, 3, 4, 5*; буква С — *2, 4, 5, 6*; буква У — *1, 3, 4, 6, 10*; буква Ч — *1, 3, 4, 10*.

ИВ-6

Индикатор вакуумный люминесцентный одноразрядный для отображения информации в виде цифр, букв, точки.
Оформление — стеклянное, сверхминиатюрное. Индикация производится через боковую поверхность баллона. Размер знакоместа 6,9×11,2 мм. Изображение формируется из светящихся анодов-сегментов. Цвет свечения — зеленый. Масса 11 г.
Pin connections *1—6, 10, 11* — anode segments *7* — cathode проводящий слои на внутренней поверхности баллона; *9* — grid (маска).



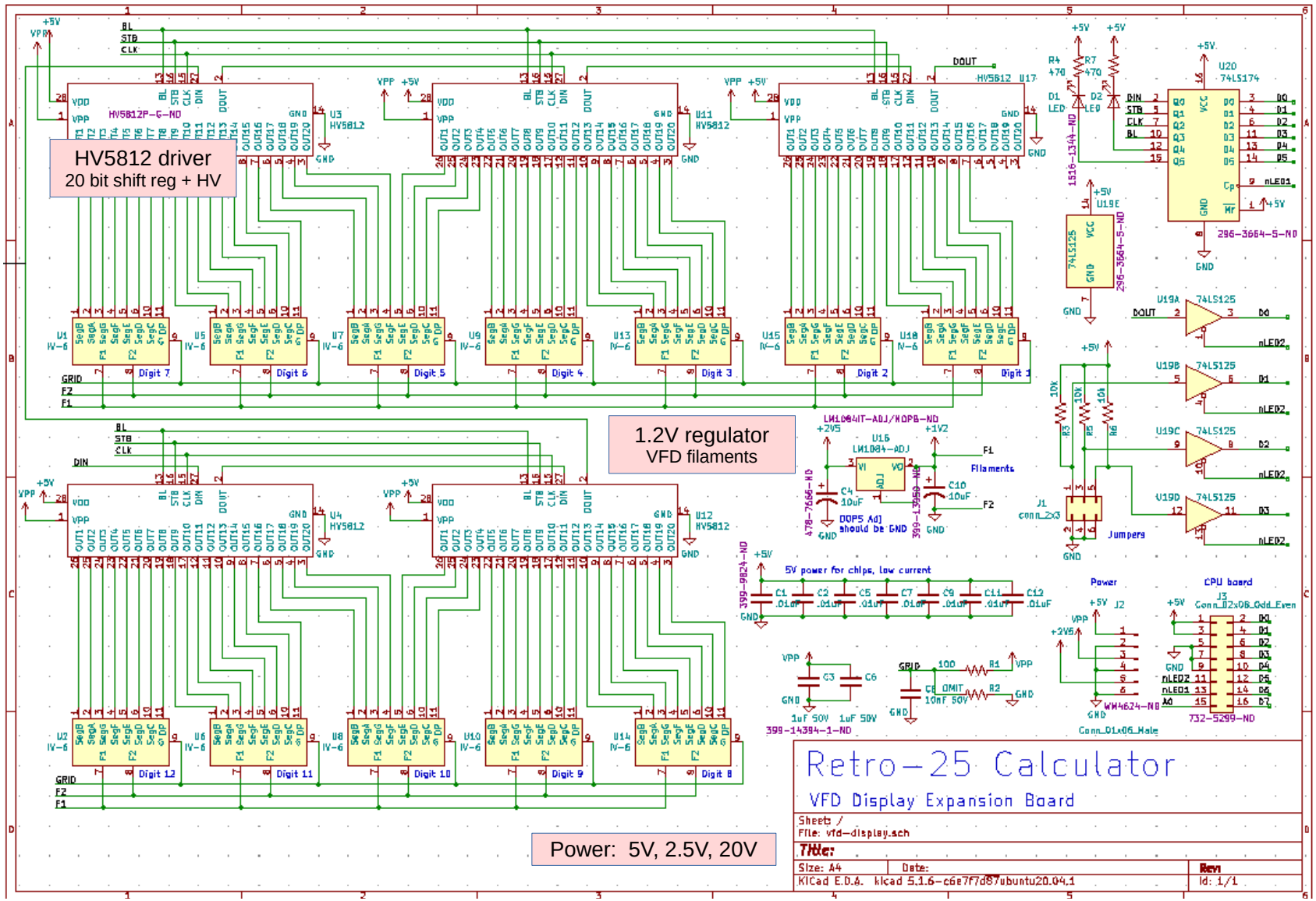
Основные данные

Brightness	300—600 кд/м²
View angle	⩾80°
Glow Current	50±5 мА
Anode segment current	
in static mode	⩽0,5 мА
in pulse mode	⩽2 мА
Anode current total	⩽1,8 мА
Grid current	
in static mode	⩽10 мА
in pulse mode	⩽45 мА
Glow voltage	$1{,}2^{+0{,}15}_{-0{,}2}$ В
Anode and grid voltage	
in static mode	25—30 В
in pulse mode	50—70 В
Operating time	⩾10 000 ч

Примечание. Для получения цифр и букв рекомендуется подключать выводы анодов следующим образом: цифра 1 — *1, 10*; цифра 2 — *1, 2, 3, 5, 6*; цифра 3 — *1, 2, 3, 6, 10*; цифра 4 — *1, 3, 4, 10*; цифра 5 — *2, 3, 4, 6, 10*; цифра 6 — *2, 3, 4, 5, 6, 10*; цифра 7 — *1, 2, 10*; цифра 8 — *1, 2, 3, 4, 5, 6, 10*; цифра 9 — *1, 2, 3, 4, 6, 10*; цифра 0 — *1, 2, 4, 5, 6, 10*; точка — *11*, буква А — *1, 2, 3, 4, 5, 10*; буква Б — *2, 3, 4, 5, 6, 10*; буква Г — *2, 4, 5*; буква Е — *2, 3, 4, 5, 6*; буква З — *1, 2, 3, 6, 10*; буква Н — *1, 3, 4, 5, 10*; буква О — *1, 2, 4, 5, 6, 10*; буква П — *1, 2, 4, 5, 10*; буква Р — *1, 2, 3, 4, 5*; буква С — *2, 4, 5, 6*; буква У — *1, 3, 4, 6, 10*; буква Ч — *1, 3, 4, 10*.

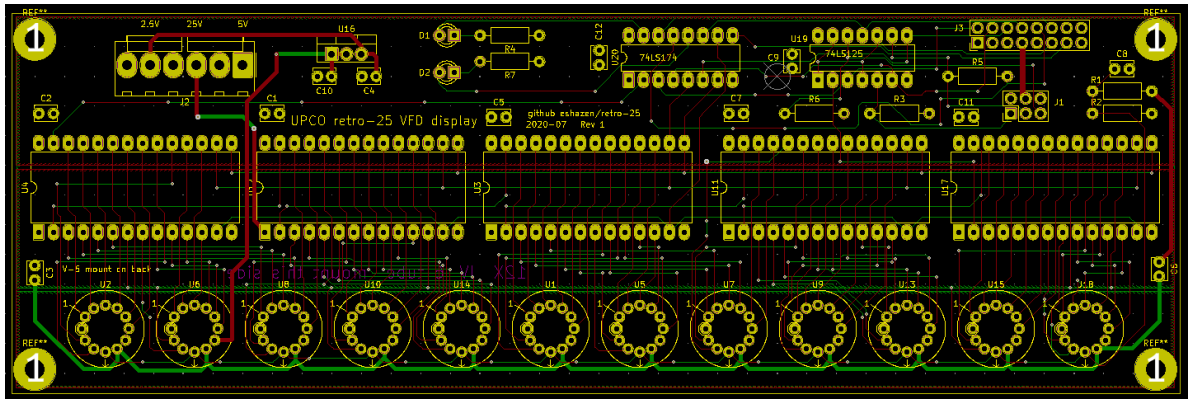
VFD Display Board

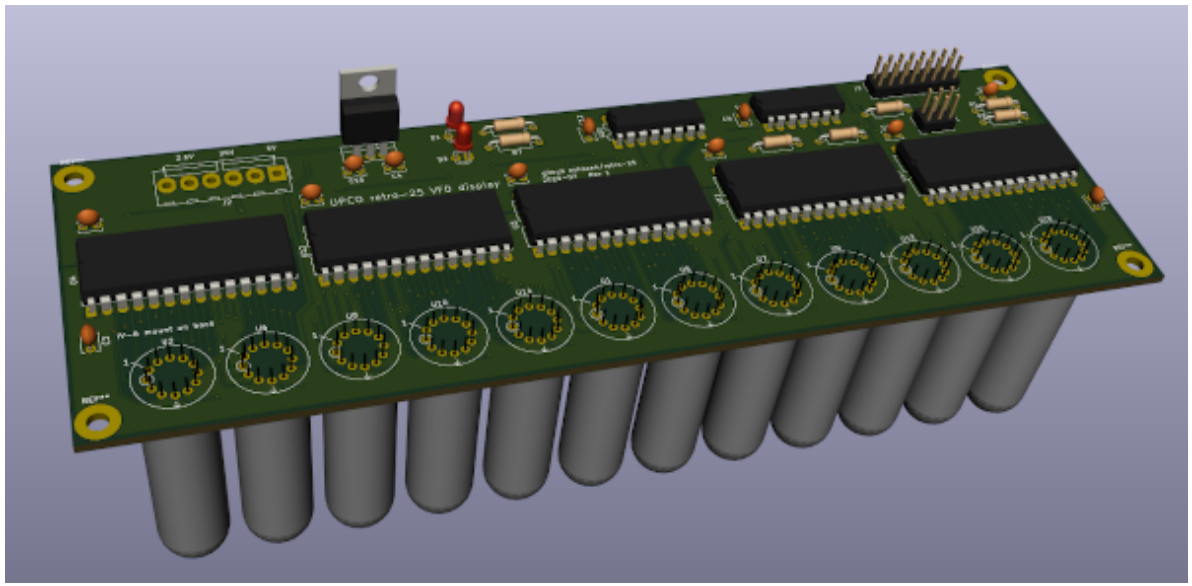
Schematic



VFD Display Board

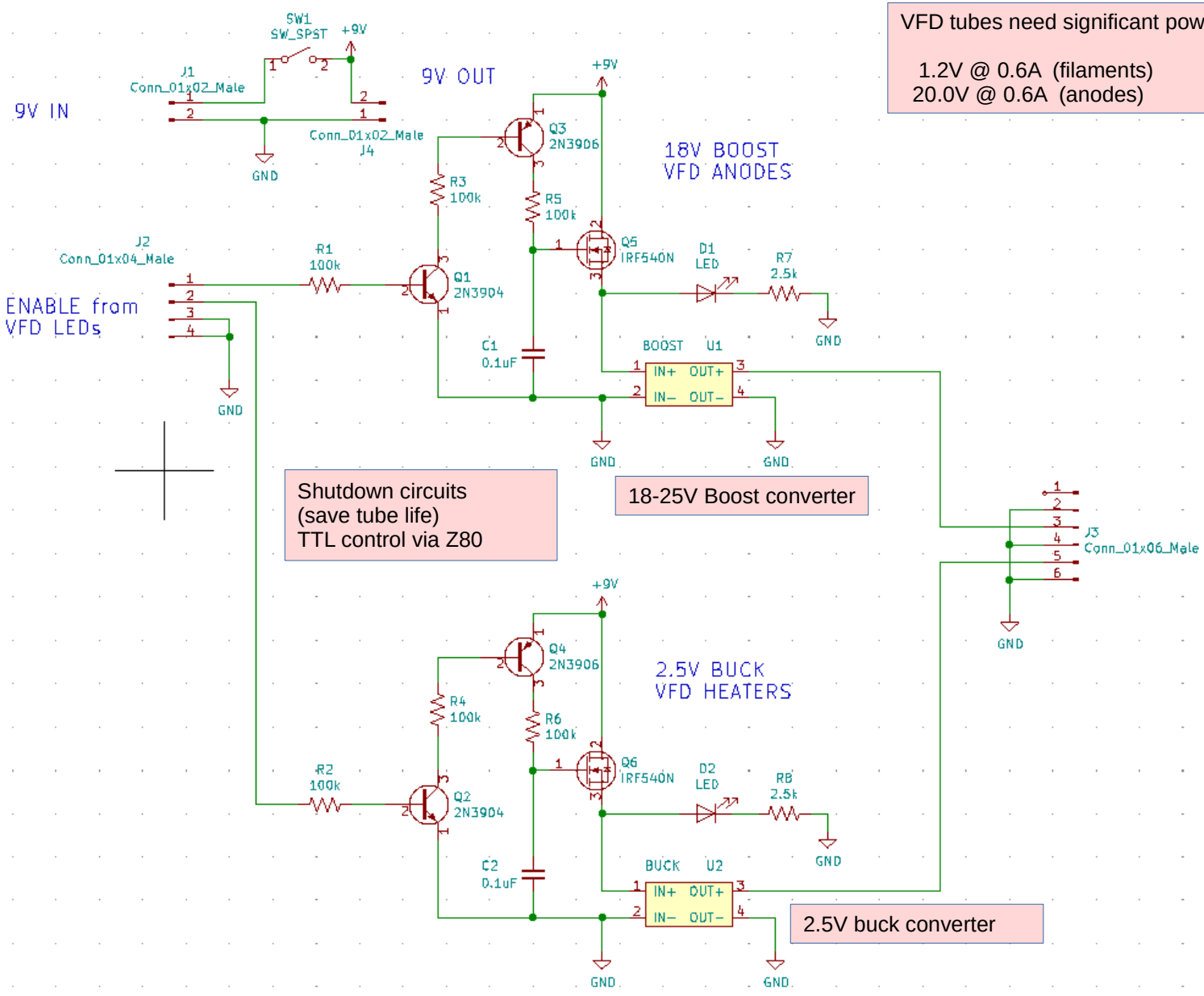
PCB Design





- 12 display tubes
- 5 HV5812 driver chips
- 2 TTL chips
- 1.2V @ 1A for filaments
- Power connector (5V, 2.5V, 20V)
- Ribbon cable compatible with LED board
- Two layer, pretty simple layout
- Fun with 3D modeling!

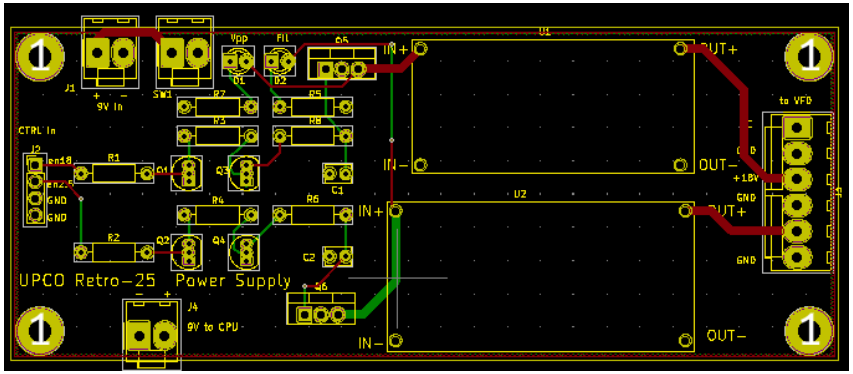
VFD Power Supply

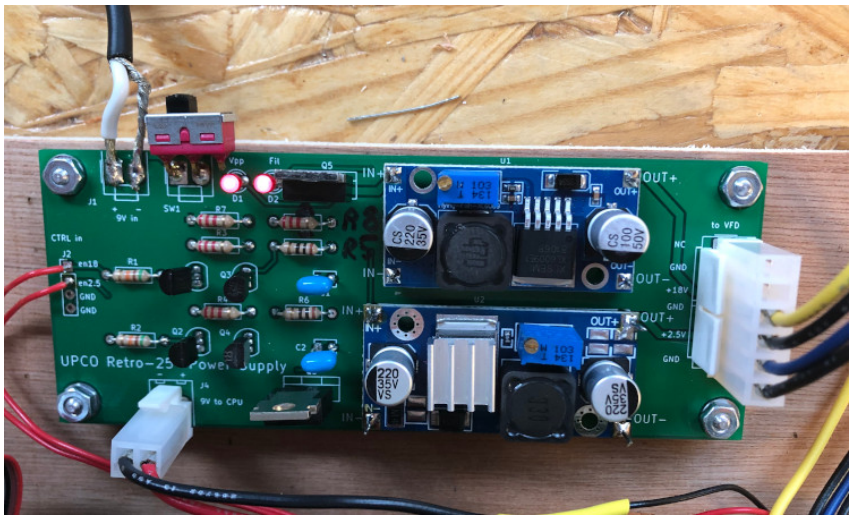


VFD tubes need significant power:

1.2V @ 0.6A (filaments)
20.0V @ 0.6A (anodes)

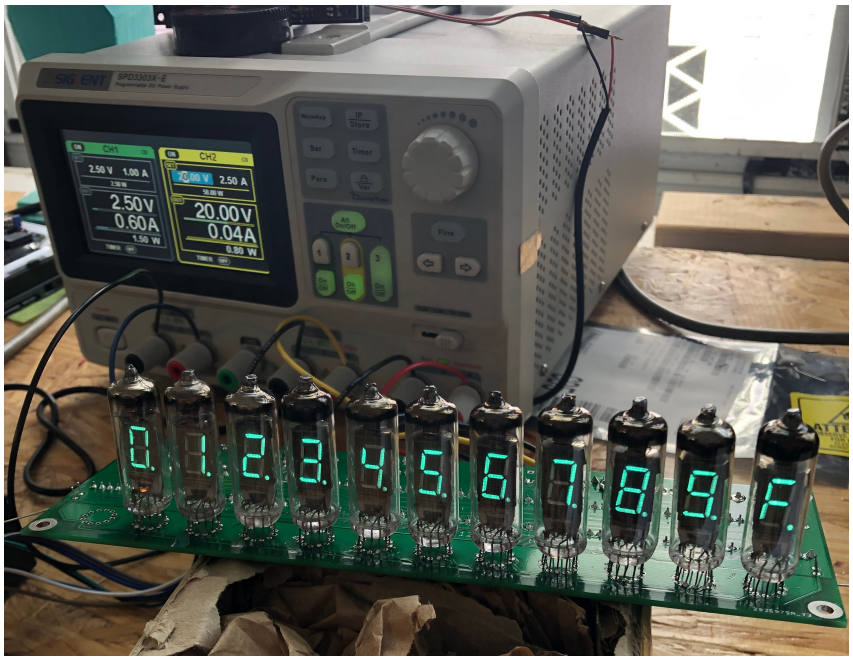
VFD Power Supply

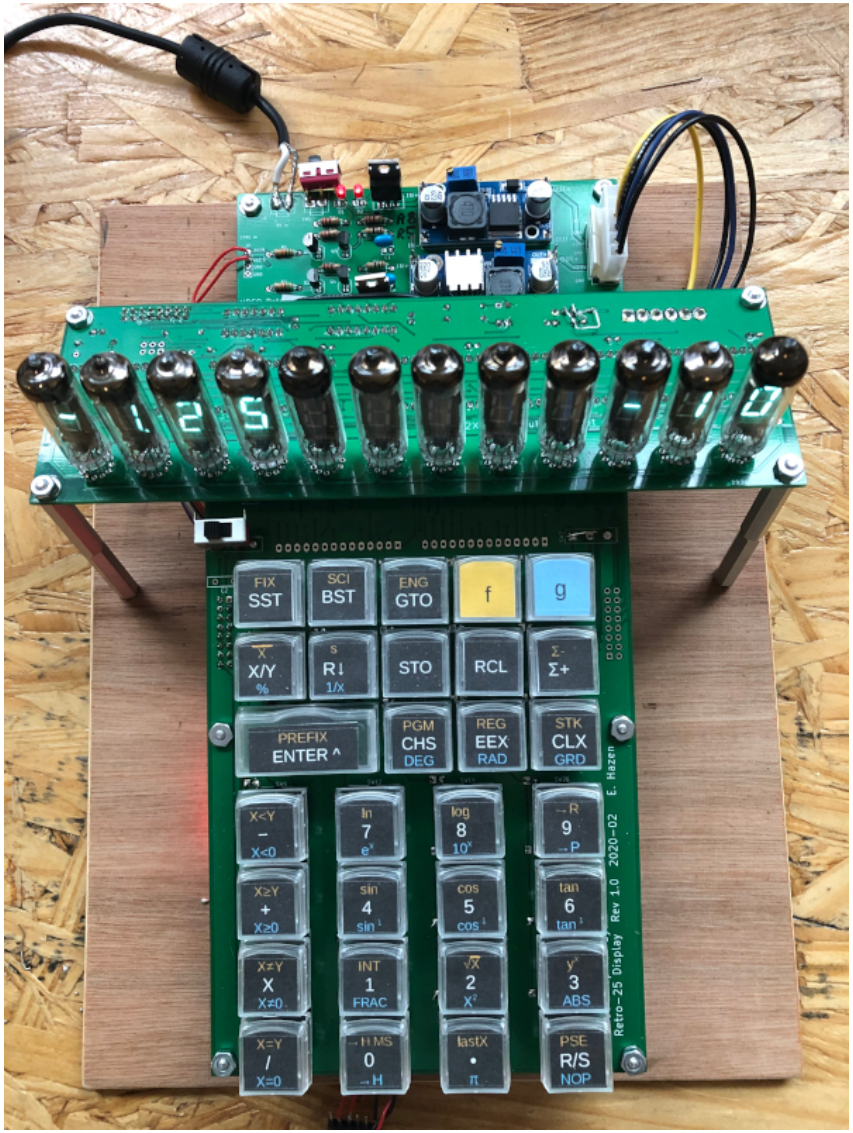




- Use Buck / Boost modules from China
- Simple switching logic to control from Z80
- Add a power switch for the whole unit

Putting it All Together



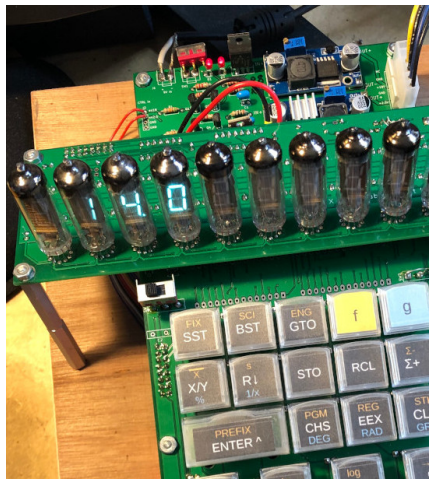


- Test the display using an Arduino
- It works!

Summary, Extras

- I've built two HP-25 replicas using Z80 CPUs
- One with 10mm LEDs, one with IV-6 VFDs
- Both run HP-25 microcode at normal speed

I currently have no plans to sell kits or finished units, but all documentation and firmware are here:
https://github.com/eshazen/retro-25



BACKUP

Extras!

Simple "assembler" and program loader

```
$./asm25.pl microstrip.txt
```

Source code entered by user

```
#
# PCB Microstrip impedance from Analog Devices MT-094
# "Microstrip and Stripline Design"
# Inputs:
#  R0 - trace width
#  R1 - trace thickness
#  R2 - height above plane
#  R3 - relative dielectric constant
# Results:
#  X  - single-ended impedance (also in R4)
#  Y  - propagation speed ps/inch
#
 1  : RCL 2      # calculate 5.98*h
 2  : 5
 3  : .
 4  : 9
 5  : 8
 6  : X
 7  : RCL 0      # calculate 0.8*w + t
 8  : .
 9  : 8
10  : *
11  : RCL 1
12  : +
```

Listing with display codes

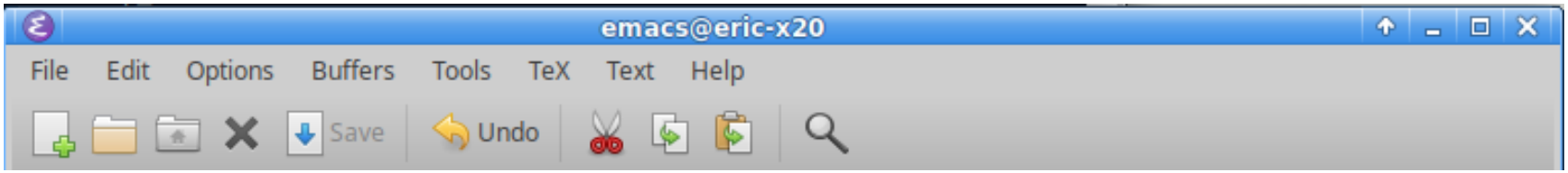
```
#
# PCB Microstrip impedance from Analog Devices MT-094
# "Microstrip and Stripline Design"
# Inputs:
#  R0 - trace width
#  R1 - trace thickness
#  R2 - height above plane
#  R3 - relative dielectric constant
# Results:
#  X  - single-ended impedance (also in R4)
#  Y  - propagation speed ps/inch
#
"24 02"    2b   1  : RCL 2      # calculate 5.98*h
"05"       c5   2  : 5
"73"       f4   3  : .
"09"       c9   4  : 9
"08"       c8   5  : 8
"61"       f2   6  : X
"24 00"    0b   7  : RCL 0      # calculate 0.8*w + t
"73"       f4   8  : .
"08"       c8   9  : 8
"61"       f2  10  : *
"24 01"    1b  11  : RCL 1
"51"       f1  12  : +
```

Loader program (via USB):

```
$./load_prog /dev/ttyUSB0 microstrip.hex
```

Extras!

Emacs integration




```
| Quantity         | Value |
|------------------+-------|
| Trace Width      |  .010 |
| Trace thickness  | .0014 |
| Dielectric       |  .008 |
| E(r)             |   4.4 |
|------------------+-------|
| Z(0)             |  58.7 |
| ps/in            | 141.2 |
|                  |       |
```

Cut/paste to/from Emacs buffer via USB interface

Example: calculate PCB trace impedance

- Perform calculation on Retro-25
- Press `F1` key on keyboard in Emacs
- Type `C-Y` to yank result to buffer

```
; copy register from retro-25 calculator with some external help
(defun yank-calc-reg (&optional regnum)
   "Yank retro-25 calculator register.  Default is X register,
    but argument 1, 2, 3 can specify Y, Z, T"
   (interactive "p")
   (kill-new (shell-command-to-string
              (format "/home/hazen/work/retro-25/util/clip_stack %s"
                      (or regnum 0)))))
(global-set-key [f1] 'yank-calc-reg)
```

Implemented as a C++ program (`clip_stack`) which can retrieve any stack level or storage register, plus a bit of Emacs lisp.