Computer architecture

S80 computer Ver. 1.0

A retro Z80 with stackable segments

Filip Pynckels

August 13, 2019
Update August 28, 2019
Update September 20, 2019

Documents:

http://users.telenet.be/pynckels/2019-3-S80-retro-Z80-computer-with-stackable-segments_ver_2-0.pdf

http://users.telenet.be/pynckels/2019-2-S80-retro-Z80-computer-with-stackable-segments_ver_1-0.pdf

Sources, schematics, printed circuit boards:

http://users.telenet.be/pynckels/2019-3-S80-retro-Z80-computer-with-stackable-segments_ver_2-0.zip

http://users.telenet.be/pynckels/2019-2-S80-retro-Z80-computer-with-stackable-segments_ver_1-0.zip

Table of contents

List of figures

List of source code

List of tables

Introduction

Welcome to this document concerning an expandable Z80 retro computer. Our goal is to explain the design, building and programming of a Z80 based computer with retro technology.

Why would one do that, we hear you ask. Well, there are multiple things that come to mind. We started our computer experience with a VIC20 (well, one of us did), in the era where Z80 and 6502 processors were state of the art. We had, since a long time, the wish to construct a computer with comparable technology ourselves. And now, the time has come.

A second element is that lately, more and more people are interested in the retro technology we count using to make our computer. More and more people wish to understand the nitty gritty details of this technology and want to play with it. This initiative permits people to play with the used technology (or to create magic smoke) using a cheap computing environment.

The third goal is to permit people to extend their computing environment with all kinds of extra functionality (think about the Internet of Things) without having to guess how to do just that. The documentation below will be as extensive as possible to permit to understand enough of the technical details to use this computer as a full-blown Internet of Things device.

A fourth goal is that, on average, beginning IT-ers have plenty of theoretical knowledge of computer architecture. However, most of them do not know how to use that knowledge to make even a simple functioning computer. So, this document will explain how a computer architecture can be designed, build, tested and programmed. As much information as possible will be added so to permit the reader to rebuild the computer at home. The information or files that cannot be joined to this document will be made available for download (see *link at the bottom of the first page* for a link). All information will be drafted to be useful in an educational setting, or for self-study. All information will be split up in independent topics. This will permit the reader to use this document as a complete tutorial, or to only read and build some of the components.

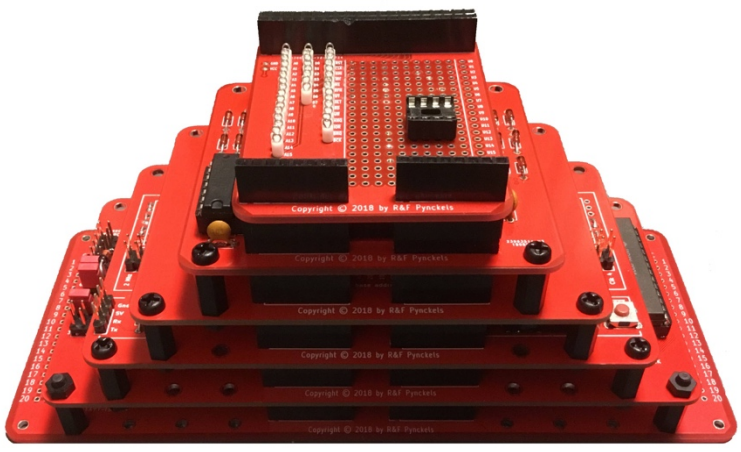

FIGURE 1 - DESIGN IMPRESSION

Everything in this document is public domain under the MIT license and can be used to develop or extend the readers projects or ideas. We would consider it an honor if people would take this

design and construct other modules for it. And we're ready to give any support possible if they want to do so.

The fact that we put a copyright notice on most documents and deliverables does not mean that we want to protect the below ideas. On the contrary, it gives us the possibility to keep this entire biotope under the MIT and the open hardware license, without someone trying to make it proprietary.

When building all the base components that are described in this document, the computer will look like the image shown in *Figure 1*. We're looking into the possibility to distribute the base components of the S80 at cost in a DIY kit (see *Figure 2*) for those readers who feel less comfortable ordering electronic components and PCB's.

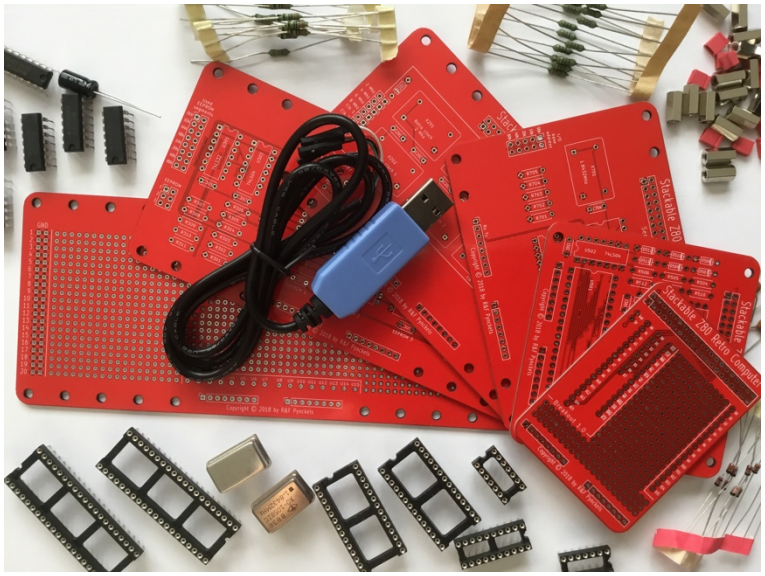

FIGURE 2 - DIY KIT

We are also looking for people who want to extend this open idea/source/hardware platform:

- Change the designs and make them faster, compacter, more elegant, more comprehensive for beginners or more challenging for experts, make them more suited for educational purposes.
- Make additional hardware/boards (CP/M 4 MB memory board, Internet of things boards, parallel I/O board, board that can be used for educational purposes, ...).
- Adapt or write software for the S80. In the first place, I think a compact monitor program would be nice. And of course, the Microsoft NASCOM ROM BASIC Ver 4.7. Maybe also some Internet of Things programs for the hardware mentioned above.
- Write educational documentation about the S80 so to permit the S80 environment to be used in schools to explain analog and digital electronics, or to be used by people interested in getting a grasp of electronics and computers.
- Host an open source version management system (independent of establishment companies) or manage a number of projects in such a system, where people can place their board designs, software developments, documentations, ... and work together on them.

- Propose other ways to make the S80 useful for people and/or society.
- Host a YouTube channel about the S80 and projects using it (tutorials on Z80, assembler, digital electronics, Internet of Things, ...).
- Promote the S80 environment as a fun and instructive environment that's entirely open hardware/software.

For the rest of this document, we will (mutatis mutandis) follow the sequence we've used to analyze, design, build and program the S80 computer, since this seems a logic way of approaching the different topics. But as said, each topic can be seen as an island on its own and serve as reference material.

At the end, a number of appendices will be added shortly. The goal is to extend this document with extra information on the functioning of the S80, on new versions of the processor, serial and memory boards, on methods the reader can use to design, build, debug and program his own S80 boards and functionality, etc.

The appendix on debugging hardware design problems will be relatively extensive with emphasis on signal measuring, logic analysis, VHDL simulation, emulation of parts of the hardware (breadboarding) and emulation of the entire hardware (wire wrapping).

1. Architecture

1.1. The Princeton architecture

The Princeton architecture (also called the Von Neuman model, see *Figure 3*) is first described in the 1945 document First Draft of a Report on the EDVAC. It contains the first published description of the logical design of a computer using the stored-program concept. The document describes a design architecture for an electronic digital computer with as components:

- A processing unit that contains an arithmetic/logic unit and processor registers
- A control unit that contains an instruction register and program counter
- Memory that stores data and instructions
- External mass storage
- Input and output mechanisms

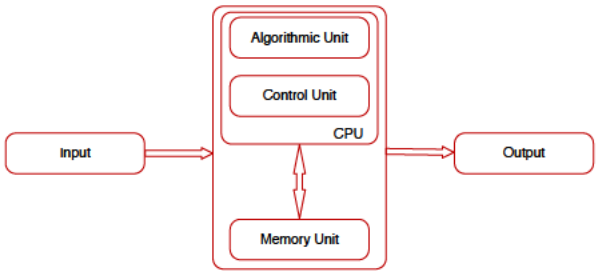

FIGURE 3 - PRINCETON ARCHITECTURE

1.2. The Harvard architecture

The Harvard architecture (see *Figure 4*) is a computer architecture with physically separate storage and signal pathways for instructions and data. The name originates from the Harvard Mark I computer. These early machines had data storage entirely contained within the central processing unit and provided no access to the instruction storage as data.

Today, most processors implement such separate signal pathways for performance reasons, but actually implement a modified Harvard architecture, so they can support tasks like loading a program from disk storage as data and then executing it.

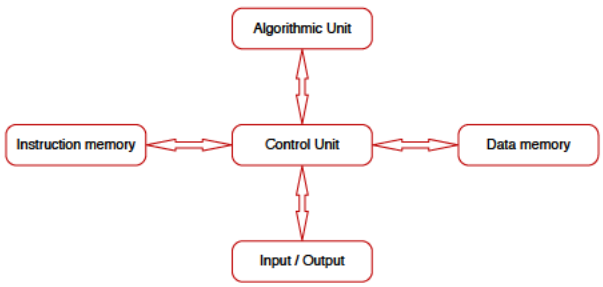

FIGURE 4 - HARVARD ARCHITECTURE

1.3. The S80 architecture

The architecture of the S80 is a combination and extension of the ideas in both the above models. We have chosen for an architecture with two busses. A primary bus with 40 lines (A, C, D, P) and a secondary bus with 16 lines (U) for later expansions.

- A[0..15] : address lines
- D[0..7] : data lines
- C[0..13] : control lines
- P[0, 1] : power lines (Gnd, Vcc)
- U[0..15] : expansion (user) lines

The number of address lines (16), data lines (8) and control lines (14) is an immediate consequence of the use of the Z80 processor and the number of lines it can manage. The number of expansion lines is an arbitrary choice, induced by design and construction choices.

A top-level schematic of the architecture can be found in *Figure 5*. The bus lines (address, data, control, power, user) are further clarified in this top-level schematic.

The processor, the memory and the serial (input/output) components are comparable with the components from The Princeton architecture. However, to make a functioning computer, a number of other functionalities have to be present. One can call these functionalities "added electronics", but I have chosen to put them on their proper component: the clock, the reset and the power component.

Comparable to the expansion lines, I have also chosen for two expansion components: a small breakout component and a large breakout component. These components permit the user to make test-components that can access the primary bus as well as the secondary bus.

The chosen architecture permits to add an unlimited number of primary/secondary bus-compatible components to be added, that can interact with already existing components by means of the bus lines. The only limitation is the inspiration of the component developer.

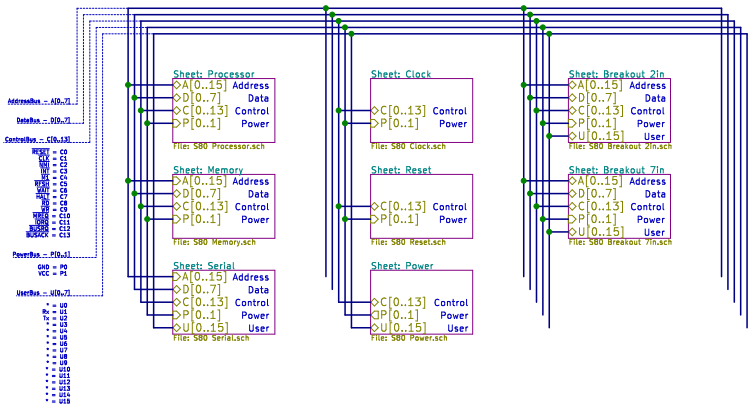

FIGURE 5 - S80 ARCHITECTURE

2. Design choices and specifications

Designing is also choosing the right tools to get the right things done in the most efficient way. In this case, the first choice to make was which tool we would use for the design of our schematics and our PCBs. We chose KiCad. It could have been another tool, and we're certainly not saying that this is the best tool, but hey, it's free and it's good.

2.1. Design guidelines

There will be 6 different board sizes (see *Figure 6*). The boards will be stackable without limit. The boards will form (eventually supported by spacers) a pyramid shape (see *Figure 1*). But, since the stacking of the boards is not limited, they can also form a diablo shape, a cube shape, a beam shape, etc. In fact, it should be elegant if not put in a case, and it should be easy to put it as compact as possible in a case for production purposes.

All components will be through-hole except if the required functionality does not exist in a DIP package. But through-hole or not, all components will be at the top side of the boards, in order to guarantee a uniform and elegant view.

Boards created by third parties can have any shape, but it would be nice if they would also consider the default measures and the default side of the board to put their components on.

The printed circuit boards will be made available in 2 colors: green and red. The red boards are the final builds of a version that are ready for "production". The green boards are testing-boards and will be marked as such. Both the green and the red boards will have white silk screens at both sides of the boards.

To guarantee that the designs stay under the MIT license, a copyright will be mentioned on the boards, so to give the designer the power to keep all his ideas in the public domain.

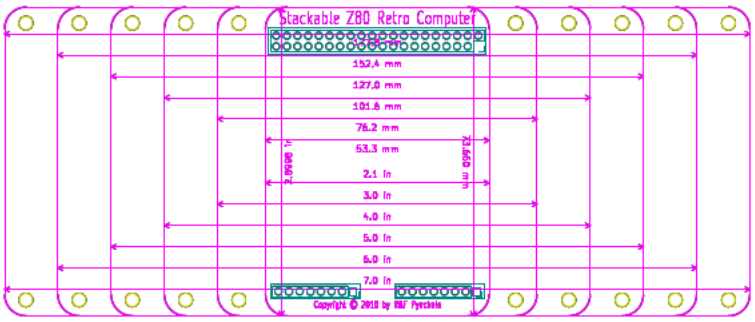

FIGURE 6 - DESIGN SPECIFICATIONS

The busses of the design will be implemented with stacking headers. The sizes of the stackable headers (A, D, C) and (U) and distance between the stackable headers (U0..U7) and (U8..U15) must permit to put a normal breadboard between two layers of the S80 as to be able to do fast temporary tests that are part breadboard and part S80. The vertical distance between any two consecutive layers of the S80 must also permit to put a standard breadboard between them (see *Figures 7 and 14 left photo*).

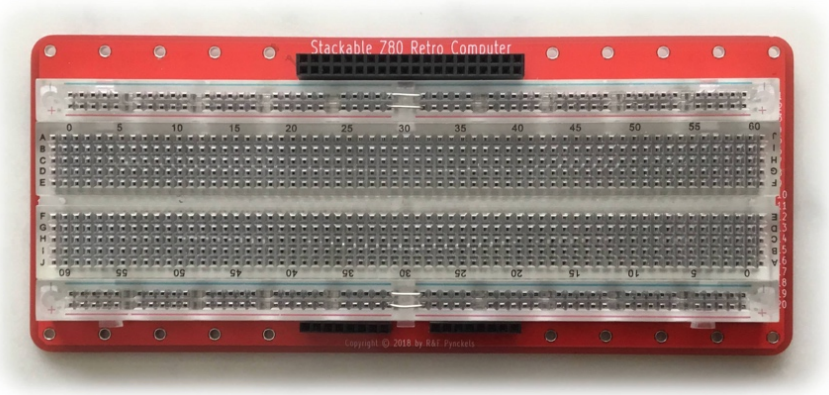

FIGURE 7 - DESIGN SPECIFICATIONS (HEADERS AND BREADBOARD)

2.2. Technical design guidelines

The technical design will be so that it gives new and/or unexperienced users a certain protection against error and magic smoke. One can think about as much copper filled areas (Vcc and Gnd) as possible, buffering of more expensive chips, current-protection, voltage clamping, etc. Each board design must go through a phase of *"user friendliness checking"* before it will be released from schematics quality control.

2.3. PCB design rules

The used PCB design rules are:

- 6 mil minimum trace width
- 6 mil minimum trace spacing
- 10 mil minimum drill size
- 5 mil annular ring
- PCB thickness of 1.6mm (0.063")
- 1oz copper on both sides (1.4 mil, 35µm)

The spacer holes must be copper surrounded and connected to the GND plane. This will guarantee that metal spacers, if used, can connect the ground planes of all boards. This is also guaranteed by the GND line of the primary bus, but the metal spacers can provide more capacitance and also a Faraday effect between the boards.

3. Clock/Power/Reset board

Most, if not all, processor IC's need at least one external clock signal to function at a predefined pace. Also, a clock is one of the easiest things to design and to debug. Just connect the test circuit to an oscilloscope or a grown-up multimeter if an oscilloscope is not available, and you can see the generated wave form, or with the multimeter, the average and the peak voltages of the wave form. The average is an indication of the high-low pulse distribution. With a high-low pulse distribution of 50%, the average voltage would be 50% of the peak-to-peak voltage difference.

Since as well the power as the reset functionality have tiny PCB dimensions, compared to the other functionalities we need for the S80, we have chosen to join the power functionality and the reset functionality with the clock functionality. Again, testing is easy, and the functionality is a prerequisite to make the other boards of the S80 computer functional.

3.1. Clock - Description

This board constitutes the heart of the S80. It is composed of multiple oscillators. The frequencies are chosen so to accommodate most of the retro Z80's that are available on the market today.

The Clock board contains 2x8 jumper pins. It will only contain one jumper. The reason we are working with jumpers, and not with dipswitches and that we only join one jumper with the board is to protect the user against himself. More specific, putting more than one jumper in place would link two or more oscillator outputs together, with an unpredictable voltage, current and wave form as a consequence.

The pins that are connected by the jumper send the signal of the corresponding oscillator to the /CLK (C1) bus. It is, however, possible to use the pins of the other oscillators to break out the oscillator signal to another board, or to the user bus. This is the reason that we include place for 4 supplementary user oscillators. This permits to concentrate all clock functionality on one board, without substantially limiting the number of available clock signals.

3.2. Power - Description

This board also constitutes the energy plant of the S80. The power functionality is composed of a simple circuit that regroups the power signals (5V and GND) and the serial IO signals. The reason is that we have opted to use, for the first version, an USB to serial TTL cable that converts the USB protocol for us. A later version will contain a USB to serial TTL circuit on-board. The only other element that is available in the power circuit is a power smoothing capacitor and a clipping Zener diode of 5v1.

3.3. Reset - Description

This board contains the reset switch for the entire configuration. Again, the PCB footprint of this functionality is so small that it doesn't motivate a separate board implementation.

The implemented reset is a normal reset as generally known and used (we provide 10ms of pulse so to give the processor at least 3 processor cycles as required in the technical specs of the Z80).

The power on reset (delayed reset when the hardware is powered on in order to let all IC's start up before the Z80) is included in the left part of the normal reset circuit.

The reset as described in the US patent 4486827 is a bit peculiar and is not implemented for the moment. The next version of the reset circuit will probably contain this reset logic.

In short: this button resets only during the first two cycles (T1 and T2) of the operation, characterized by /M1 being low. The goal of this reset is to use the same /RESET control input to the processor as a normal reset, to reset only the program counter to facilitate development (breakpoints)."

3.4. Clock/Power/Reset - Technical specifications

Table 1 gives an overview of most of the information needed to integrate the clock/power/reset board in the S80 stack.

Author/Creator	Robin & Filip Pynckels
License	MIT License (open hardware and software)
Remark	The clock logic is regrouped with the power and the reset logic on one board.
Status	Production boards are ready. Do it yourself kits are looked into.
PCB Manufacturing	JLCPCB batch numbers: 2396351A-Y4-180909
Dimensions (W x H x D)	127.0mm (5.0") x 73.7mm (2.9") x 1.6mm (0.063")
Outgoing bus signals	C[0, 1] (non-buffered, /RESET and /CLK) P[0, 1] (no protection build in, GND and VCC, incoming on connector)
Incoming bus signals	-
Bidirectional bus signals	-
User bus signals	U1 (non-buffered, outgoing on bus and incoming on connector, Rx) U2 (non-buffered, incoming on bus and outgoing on connector, Tx)
Reserved Memory addresses	-
Reserved IO addresses	-

TABLE 1 - CLOCK/POWER/RESET BOARD - TECHNICAL SPECIFICATIONS

3.5. Clock - Functional schematics

A full-size version of all the functional schematics can be found in *Appendix 1*. To give an overview however, we join a rescaled version of the functional schematics of the clock in *Figure 8*.

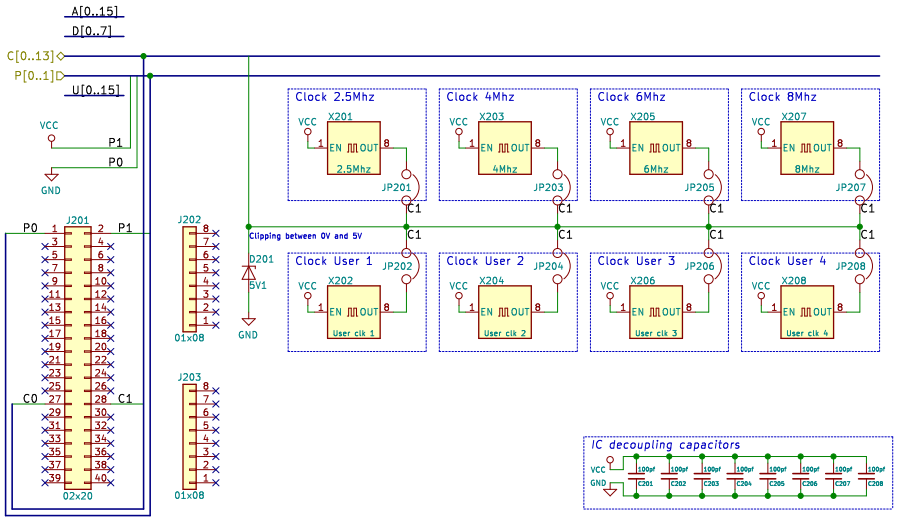

FIGURE 8 - CLOCK FUNCTIONAL SCHEMATICS

3.6. Power - Functional schematics

A full-size version of all the functional schematics can be found in *Appendix 1*. To give an overview however, we join a rescaled version of the functional schematics of the power circuit in *Figure 9*.

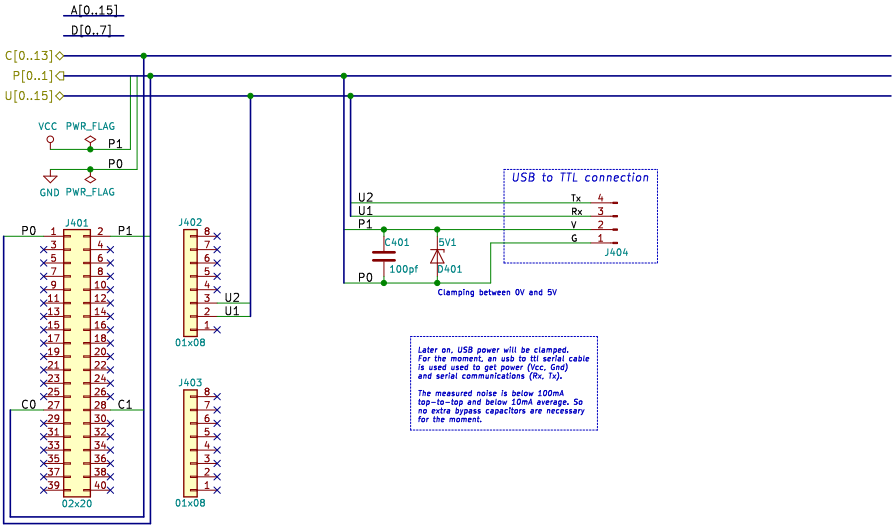

FIGURE 9 - POWER FUNCTIONAL SCHEMATICS

3.7. Reset - Functional schematics

A full-size version of all the functional schematics can be found in *Appendix 1*. To give an overview however, we join a rescaled version of the functional schematics of the power circuit in *Figure 10*.

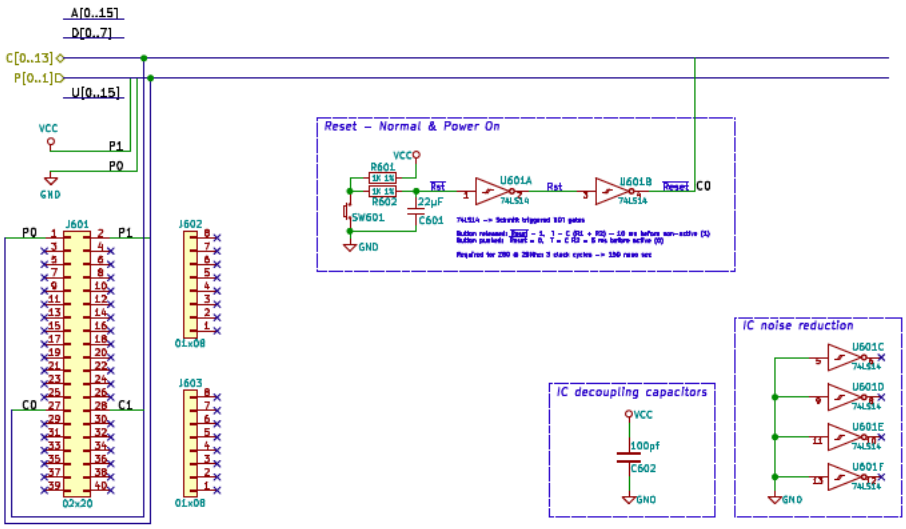

FIGURE 10 - RESET FUNCTIONAL SCHEMATICS

3.8. Cock/Power/Reset - PCB design

A real-size version of the PCB design can be found in *Appendix 2*. To give an overview however, we join a rescaled version of the PCB design of the clock in *Figure 11*. An impression of the non-populated final PCB is given in *Figure 12*. Please note that the below figures don't have the exact dimensions of the real PCB's

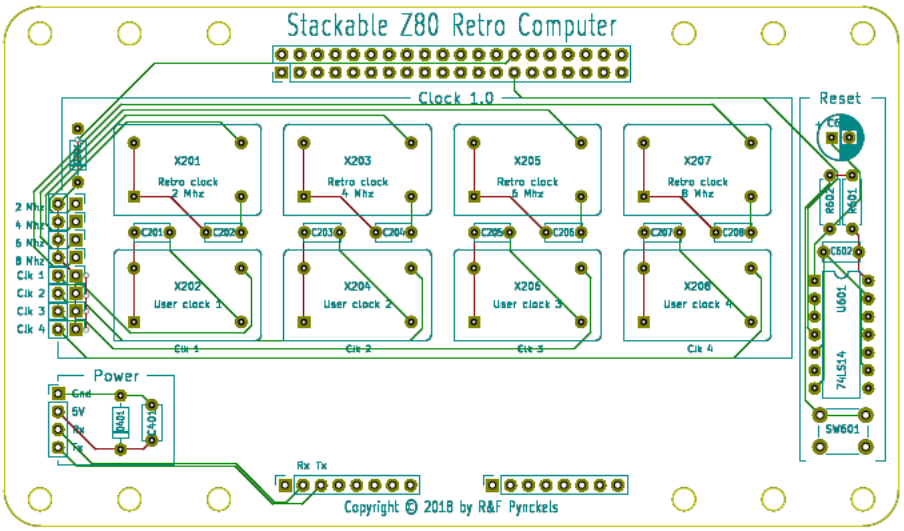

FIGURE 11 - CLOCK/POWER/RESET PCB DESIGN

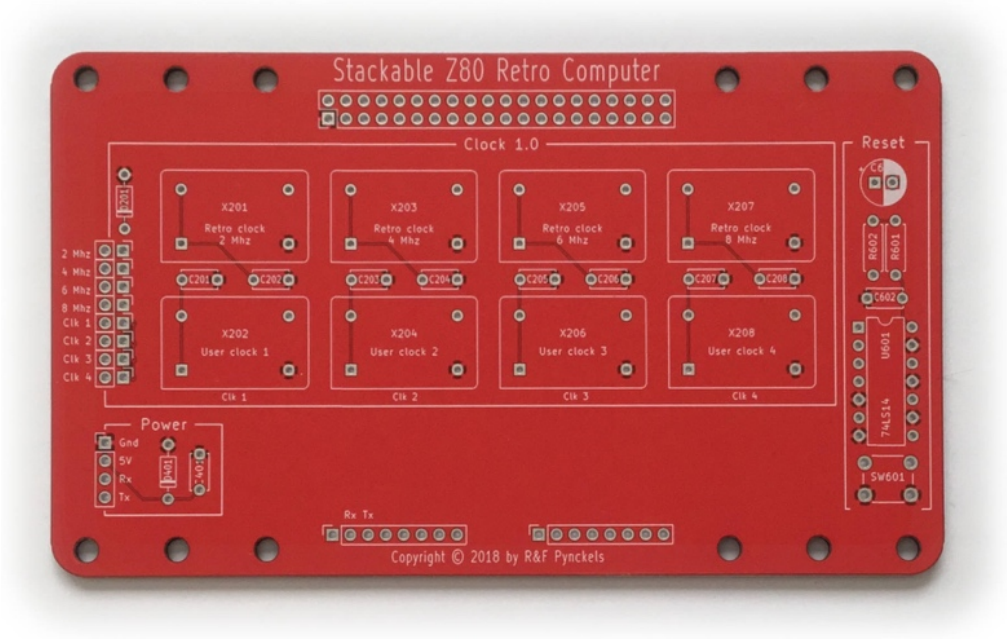

FIGURE 12 - CLOCK/POWER/RESET PCB

3.9. Reset - Timing tests

The reset circuit resets on demand and does a delayed reset on power on (see *Figure 13* for timing results). It should provide the special debugging reset, but that feature is not implemented yet.

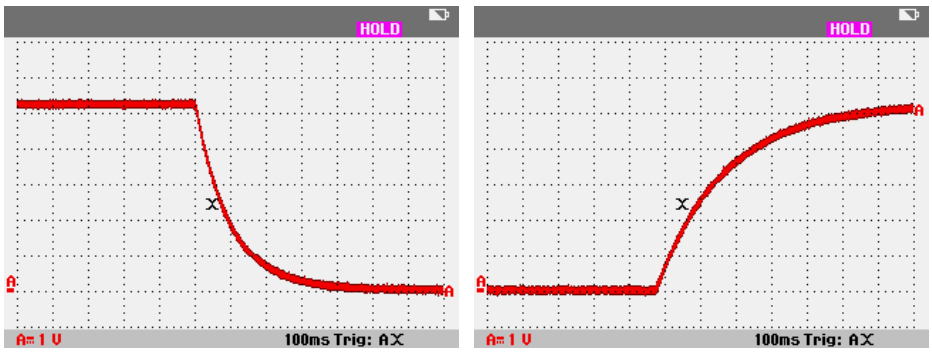

FIGURE 13 - RESET TIMING RESULTS

4. Processor board

4.1. Processor - Description

The next logic step in the design and construction of the S80 computer is the processor functionality. We can use the tested clock, power and reset functionality to drive the processor board (test) circuits.

The first functional schematics will contain a number of protection elements. More specific, we provide power peak protection and the necessary pull-up resistors to guarantee that the processor will not be blown up by errors made on other boards and that the processor will not be connected to address, control, data or user lines that have the wrong logic level.

Extensive testing will be done to assess the correct functioning of this board before declaring it "production ready" (see *Figure 14*).

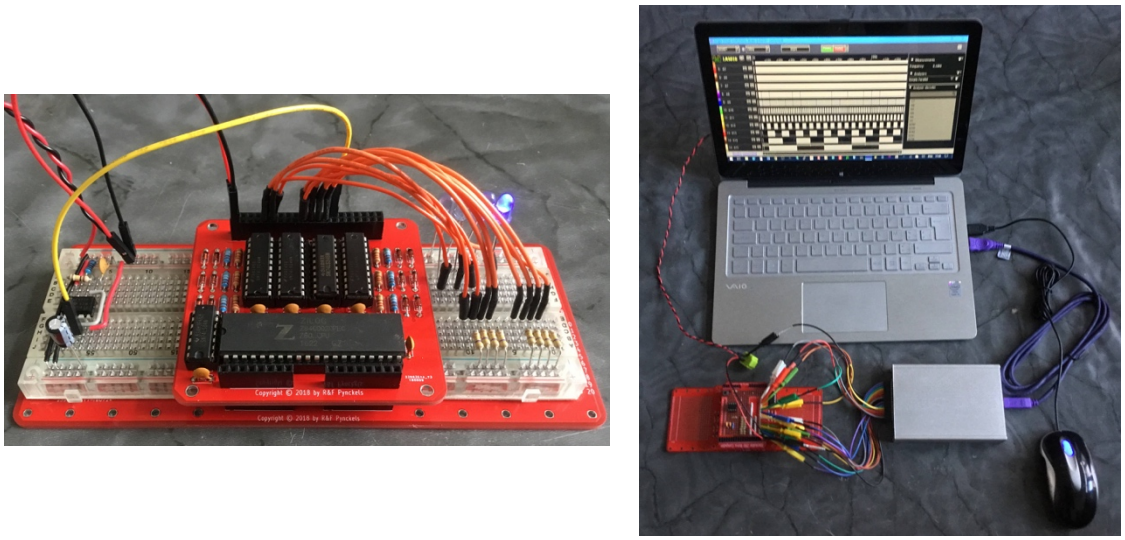

FIGURE 14 - TEST CONFIGURATIONS

4.2. Processor - Technical specifications

Table 2 gives an overview of most of the information needed to integrate the processor board in the S80 stack.

Author/Creator	Robin & Filip Pynckels
License	MIT License (open hardware and software)
Status	Production boards are ready. Do it yourself kits are looked into.
PCB Manufacturing	JLCPCB batch numbers: 2396351A-Y3-1800809
Dimensions (W x H x D)	76.2mm (3.0") x 73.7mm (2.9") x 1.6mm (0.063")
Outgoing bus signals	A[0..15] (buffered, can go in high impedance) C[4, 7..11] (buffered, can go in high impedance)

	C5, C13 (non-buffered, no high-impedance mode)
Incoming bus signals	C[0..3, 6, 12] (clamped, accept high impedance mode) P[0, 1] (no protection build in)
Bidirectional bus signals	D[0..7] (buffered)
User bus signals	-
Reserved Memory addresses	-
Reserved IO addresses	-

TABLE 2 - PROCESSOR BOARD - TECHNICAL SPECIFICATIONS

4.3. Processor - Functional schematics

To give an overview, we join a rescaled version of the functional schematics of the processor circuit in *Figure 15*.

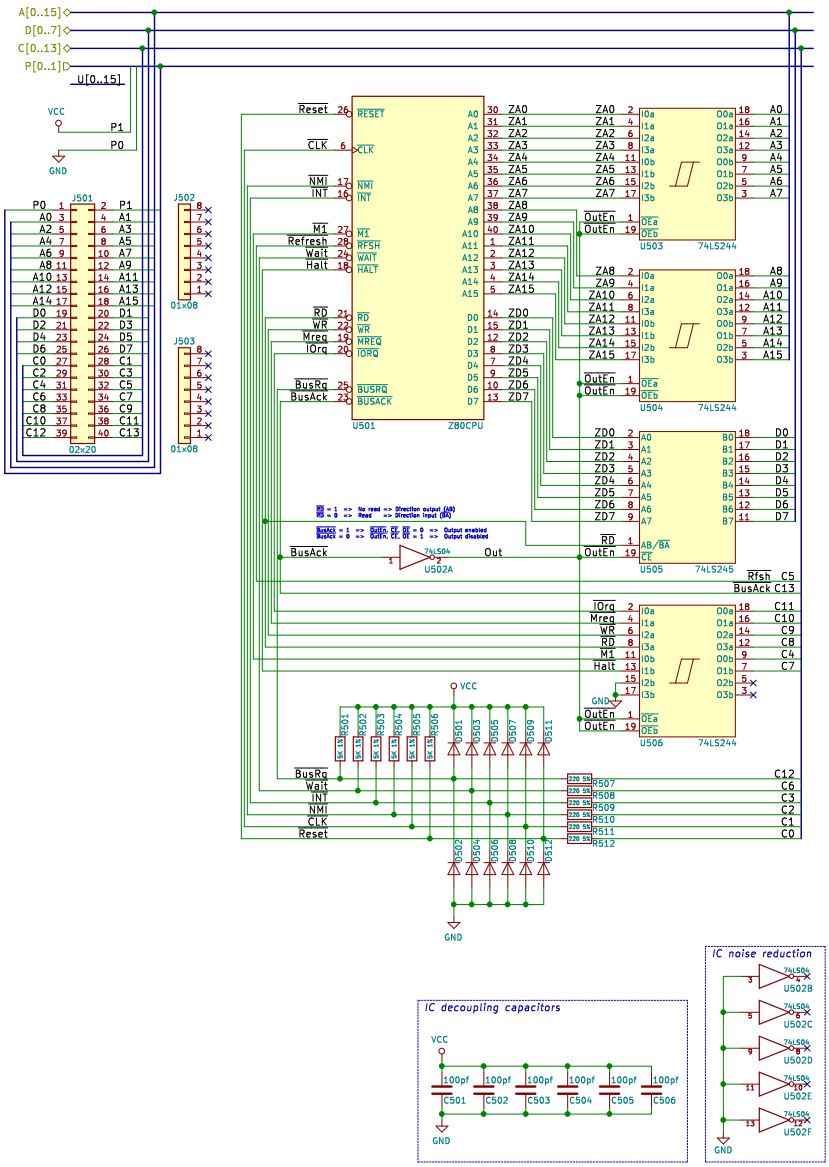

FIGURE 15 - PROCESSOR FUNCTIONAL SCHEMATICS

4.4. Processor - PCB design

To give an overview, we join a rescaled version of the PCB design of the processor board in *Figure 16*. An impression of the non-populated and the populated final PCB is given in *Figure 17*. Please note that the below figures don't have the exact dimensions of the real PCB's

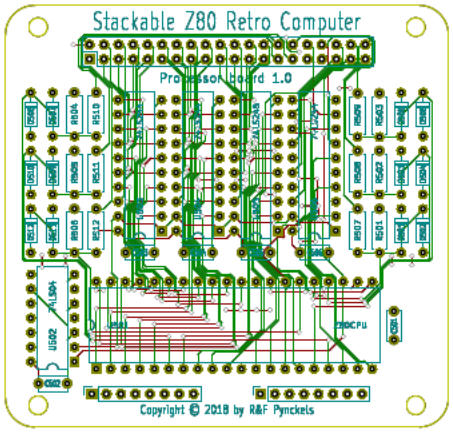

FIGURE 16 - PROCESSOR PCB DESIGN

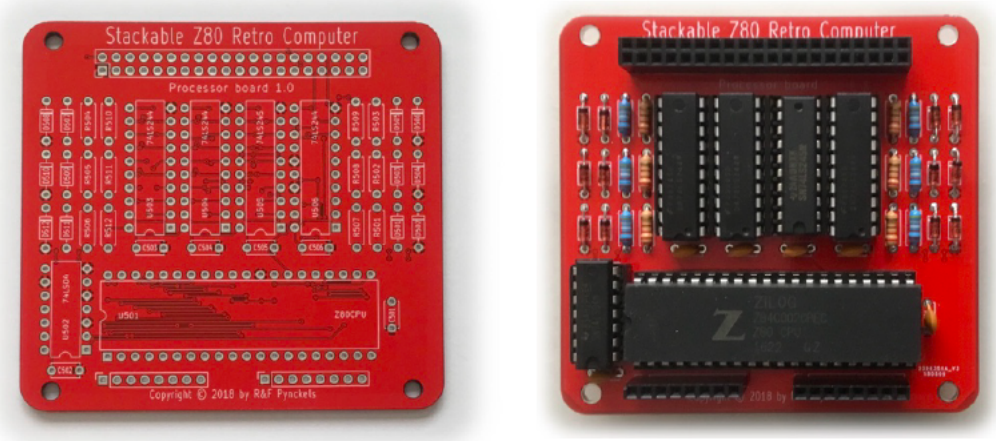

FIGURE 17 - PROCESSOR PCB

5. Serial board

5.1. Serial - Description

This board constitutes the senses of the S80. It is composed of a, 16C550 UART chip, together with some logic circuits to interact with the address bus, the data bus and the control bus.

The incoming low address byte is compared with an address set by jumpers. It should be equal. It is advised to set the address with the jumpers to 0xC0. This will use the IO address range 0xC0 to 0xC7. Another address range can be used, although care should be taken not to interfere with the IO addresses of other boards.

Now that we have a processor board, we can put little test programs on an EEPROM chip to test the serial (test) circuits (see *Figure 18*). The only thing that has the be done is connect the necessary pins of the memory chip to the processor board. Plugging the EEPROM in a breadboard and using jumper wires will do the trick.

FIGURE 18 - SERIAL TEST CONFIGURATION

5.2. Serial - Technical specifications

Table 3 gives an overview of most of the information needed to integrate the serial board in the S80 stack.

Author/Creator	Robin & Filip Pynckels
License	MIT License (open hardware and software)
Status	Production boards are ready. Do it yourself kits are looked into.
PCB Manufacturing	JLCPCB batch numbers: 2396351A-Y6-180909

Dimensions (W x H x D)	101.6mm (4.0") x 73.7mm (2.9") x 1.6mm (0.063")
Outgoing bus signals	C3
Incoming bus signals	A[0..7] (checked on address range set by dipswitches) C0, C4, C8, C9, C11 (no protection build in) P[0, 1] (no protection build in)
Bidirectional bus signals	D[0..7] (buffered)
User bus signals	U1 (Rx incoming on U-bus & TTL conn pin 3) U2 (Tx outgoing on U-bus & TTL conn pin 4)
Reserved Memory addresses	-
Reserved IO addresses	Variable base address in multiples of 0x08. Advised addresses are 0xC0 (base address) :: 0xC7

TABLE 3 - SERIAL BOARD - TECHNICAL SPECIFICATIONS

5.3. Serial - Functional schematics

To give an overview, we join a rescaled version of the functional schematics of the serial circuit in *Figure 19*.

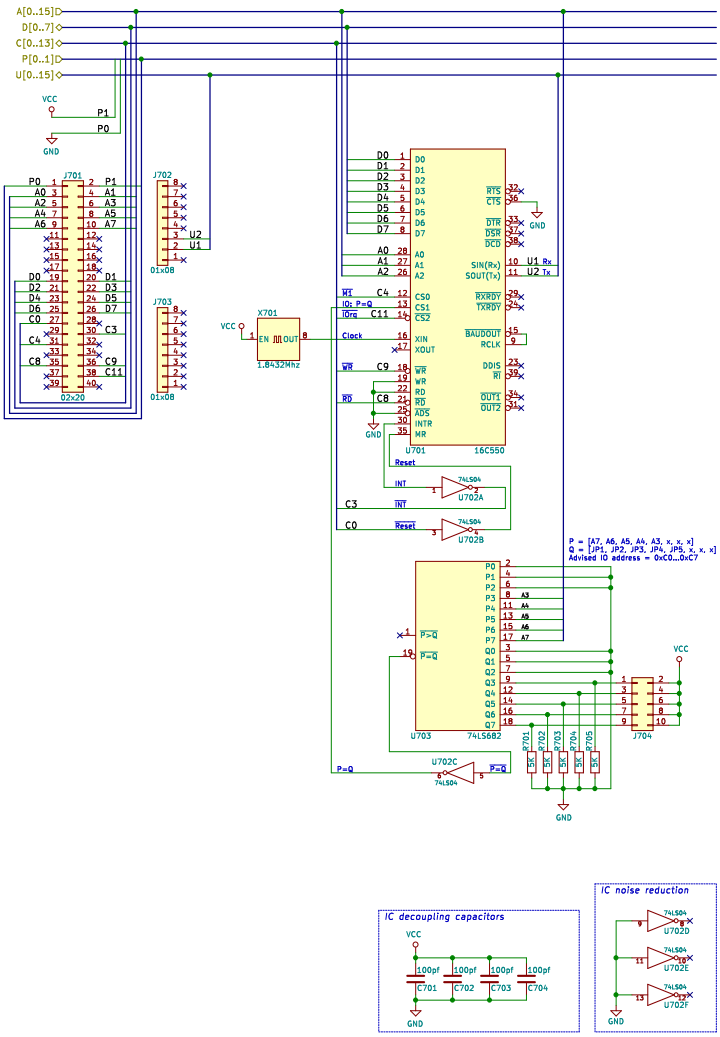

FIGURE 19 - SERIAL FUNCTIONAL SCHEMATICS

5.4. Serial - PCB design

To give an overview, we join a rescaled version of the PCB design of the serial board in *Figure 20*. An impression of the non-populated and the populated final PCB is given in *Figure 21*. Please note that the below figures don't have the exact dimensions of the real PCB's.

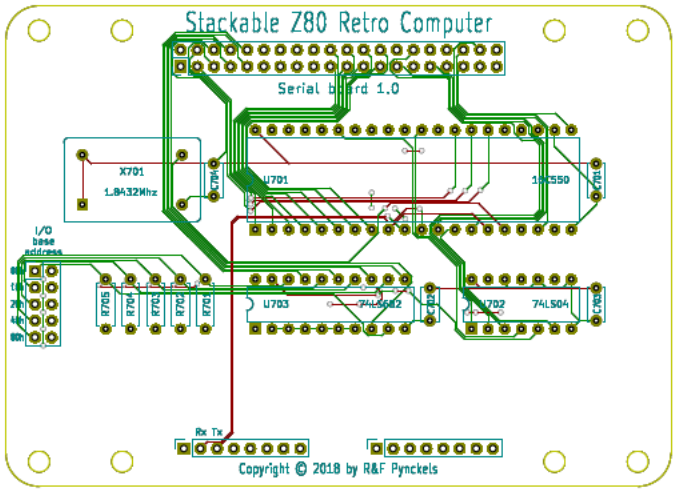

FIGURE 20 - SERIAL PCB DESIGN

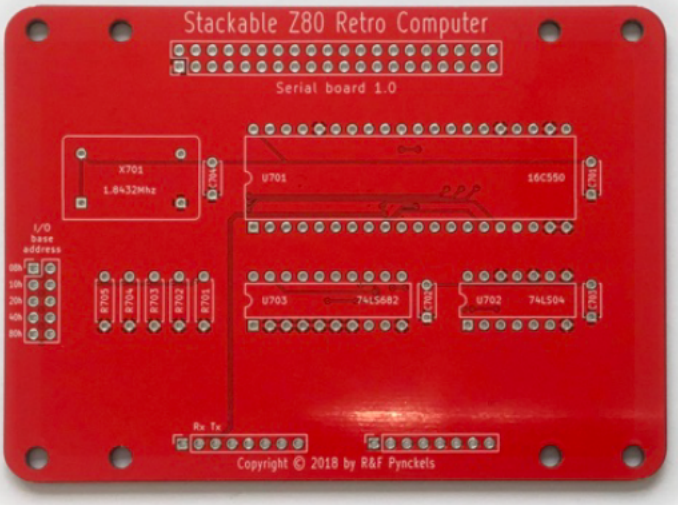

FIGURE 21 - SERIAL PCB

6. Memory board

6.1. Memory - Description

Now that we are sure about the good functioning of the processor board (and hence the clock/power/reset board) and the serial board, we can move on to the construction of the third functionally necessary board for our S80 architecture: the memory board.

This board constitutes the short- and long-term memory of the S80. It is composed of one or more EEPROM's (electrically erasable programmable read-only memory) and two static RAM chips. Together with some logic circuits to interact with the address bus, the data bus and the control bus.

The incoming signals are not clipped nor clamped, so care should be taken not to pass the 5 volts limit. Nor should a current peak occur in one of the busses when this memory board is used.

The Memory board contains 2x8 pins for jumpers. These serve to provide as much RAM as possible, without blocking the number of EEPROM bytes that can be used. More specific, this board permits to use between 0 and 32Kb of EEPROM bytes. The dipswitches permit to set the number of used 128 bytes EEPROM segments (starting from address 0x0000). RAM starts immediately after the last EEPROM segment as defined by the dipswitches. It is possible to use no EEPROM bytes. In this case, all jumpers must be disconnected (set to 0).

Another 2x3 male header permits to choose between one of three EEPROM's.

6.2. Memory - Technical specifications

Table 4 gives an overview of most of the information needed to integrate the memory board in the S80 stack.

Author/Creator	Robin & Filip Pynckels
License	MIT License (open hardware and software)
Status	Production boards are ready. Do it yourself kits are looked into.
PCB Manufacturing	JLCPCB batch numbers: 2396351A-Y5-180909
Dimensions (W x H x D)	152.4mm (6.0") x 73.7mm (2.9") x 1.6mm (0.063")
Outgoing bus signals	-
Incoming bus signals	A[0..15] (non-clamped, accepts high impedance mode if C[8, 9, 10] are not floating) C[8, 9, 10] (non-clamped, does not accept high impedance mode) P[0, 1] (no protection build in)
Bidirectional bus signals	D[0..7] (non-buffered, non-clamped, can go in high impedance mode, accepts high impedance mode if C[8, 9, 10] are not floating)
User bus signals	-
Reserved Memory addresses	0x0000 :: 0xFFFF
Reserved IO addresses	-

TABLE 4 - MEMORY BOARD - TECHNICAL SPECIFICATIONS

6.3. Memory - Functional schematics

To give an overview, we join a rescaled version of the functional schematics of the memory circuit in *Figure 22*.

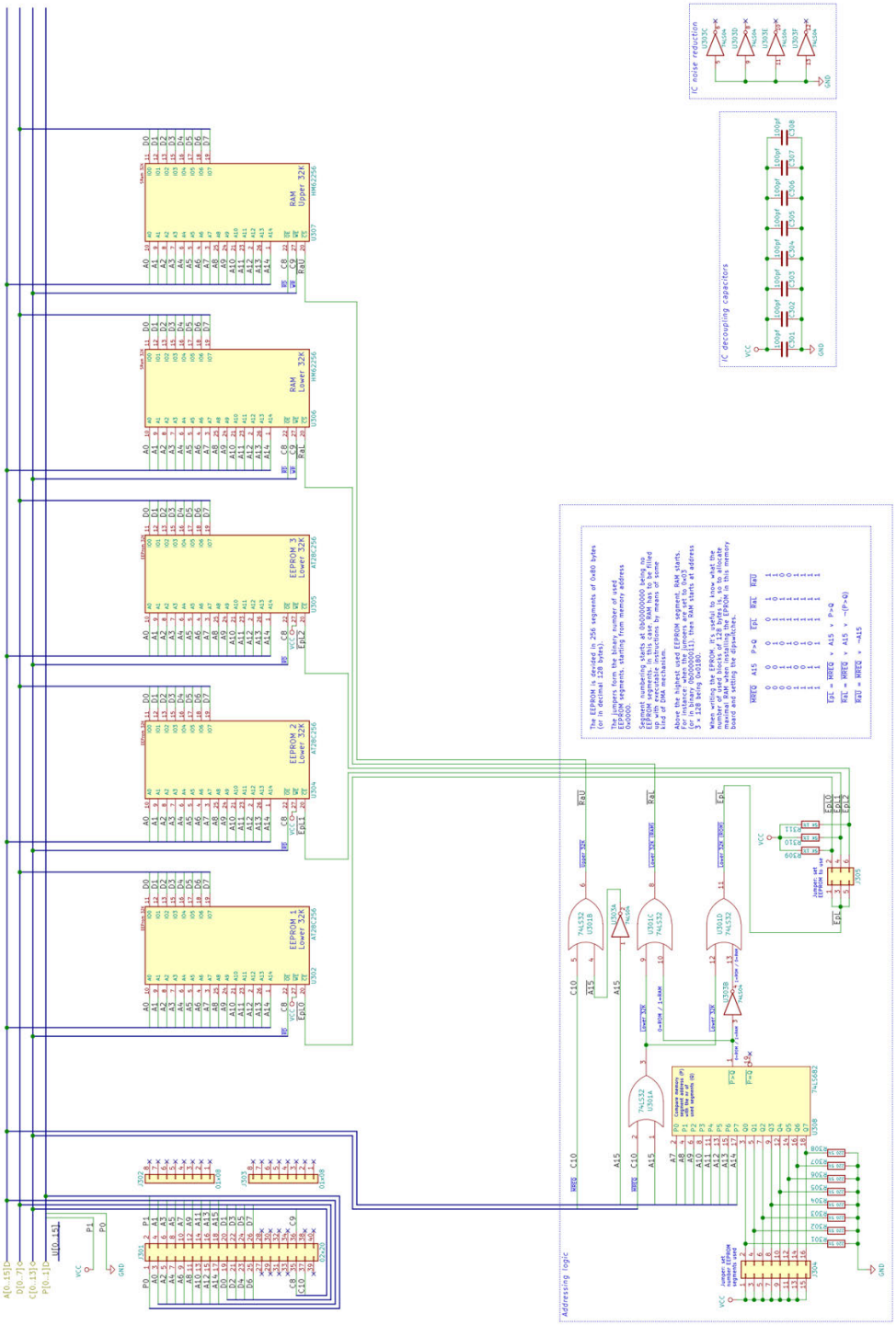

FIGURE 22 - MEMORY FUNCTIONAL SCHEMATICS

6.4. Memory - PCB design

To give an overview, we join a rescaled version of the PCB design of the memory board in *Figure 23*. An impression of the non-populated final PCB is given in *Figure 24*. Please note that the below figures don't have the dimensions of the real PCB's

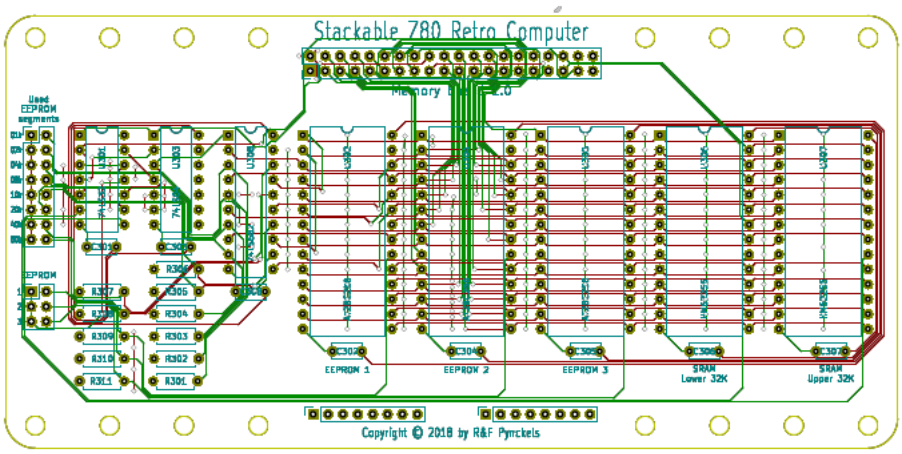

FIGURE 23 - MEMORY PCB DESIGN

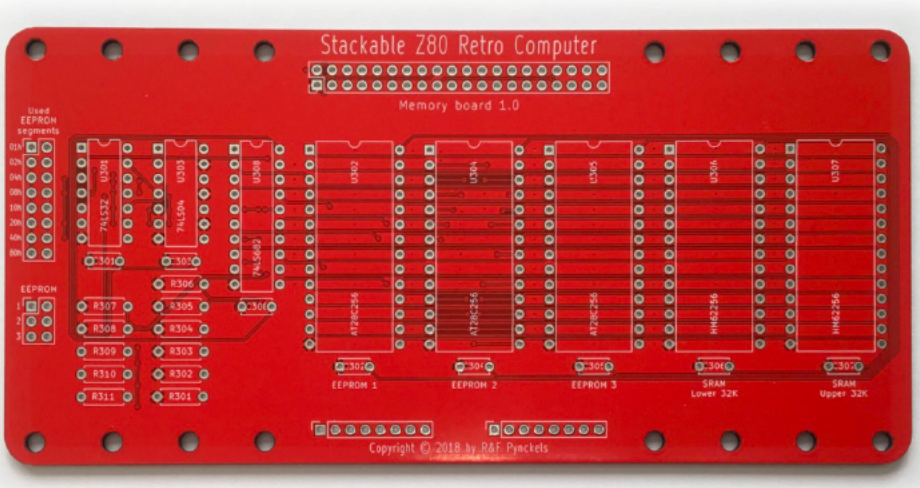

FIGURE 24 - MEMORY PCB

7. Breakout boards

7.1. Breakout - Description

These boards are the top and bottom plates of the S80 computer configuration. At the same time, they are the smallest and largest boards and give the user the possibility to append test circuits to the S80, since all busses are broken out on these boards, without any further protection or precautions. This gives the user full control over the entire S80 configuration and all its busses, memory addresses, IO addresses, etc. Care has been taken that even a 40-pin DIP socket can be placed on the smallest of the two boards, together with some other components.

For instance, *Figure 25* shows the board that can be used to do a base test of the processor board (see *Figure 14 right photo*). This particular test circuit returns a reset signal when the button is pushed, a clock signal generated by a 555 IC and data lines D[0..7] = 0, which permits to keep the Z80 on the processor board in an eternal loop. This in turn can be measured with an oscilloscope for low level debugging of the hardware and with a protocol analyzer for high level debugging of the hardware.

Note that he breakout boards are not suited for wire-wrapping. The diameter of the pads is only 0.8mm (0.031") which makes them not suited for wire-wrapping sockets, but snugger for through hole components.

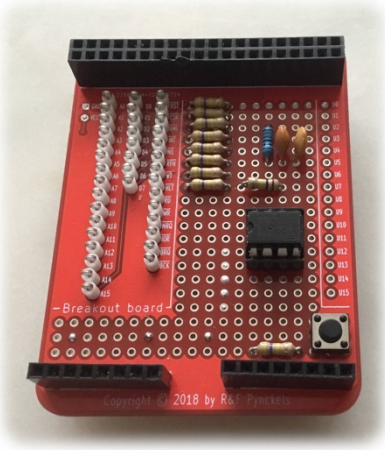

FIGURE 25 - 2" BREAKOUT WITH COMPONENTS

7.2. Breakout - Technical specifications

Table 5 gives an overview of most of the information needed to integrate the breakout boards in the S80 stack.

Author/Creator	Robin & Filip Pynckels
License	MIT License (open hardware and software)
Status	Production boards are ready. Do it yourself kits are looked into.
PCB Manufacturing	JLCPCB batch numbers: 2396351A-Y2-180724 (2 inch) 2396351A-Y1-180724 (7 inch)

Dimensions (W x H x D)	53.3mm (2.1") x 73.7mm (2.9") x 1.6mm (0.063") 177.8mm (7.0") x 73.7mm (2.9") x 1.6mm (0.063")
Outgoing bus signals	C3
Incoming bus signals	-
Bidirectional bus signals	A[0..15] (no protection build in) D[0..7] (no protection build in) C[0..13] (no protection build in) P[0, 1] (no protection build in)
User bus signals	U[0..15] (no protection build in)
Reserved Memory addresses	-
Reserved IO addresses	-

TABLE 5 - BREAKOUT BOARDS - TECHNICAL SPECIFICATIONS

7.3. Breakout - Functional schematics

To give an overview, we join a rescaled version of the functional schematics of the breakout boards in *Figures 26*. Note that the functional schematic is the same for both the boards. The PCB's are different of course.

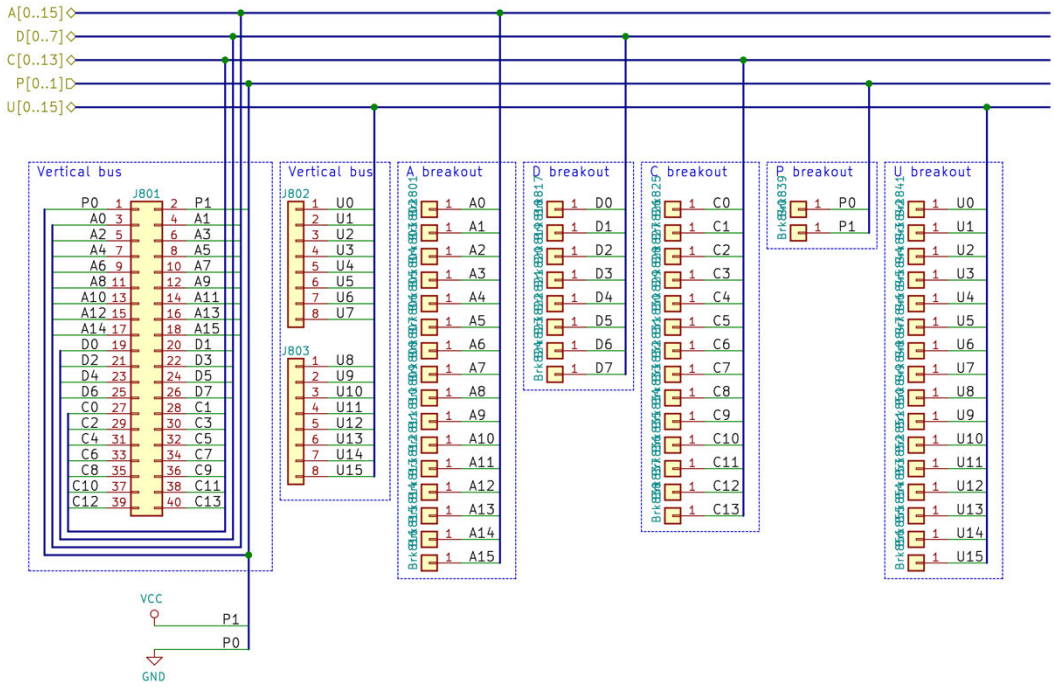

FIGURE 26 - BREAKOUT FUNCTIONAL SCHEMATICS

7.4. Breakout - PCB design

To give an overview, we join a rescaled version of the PCB designs of the breakout boards in *Figure 27*. An impression of the non-populated and the populated final PCB's is given in *Figures 28 and 29*. Please note that the below figures don't have the dimensions of the real PCB's.

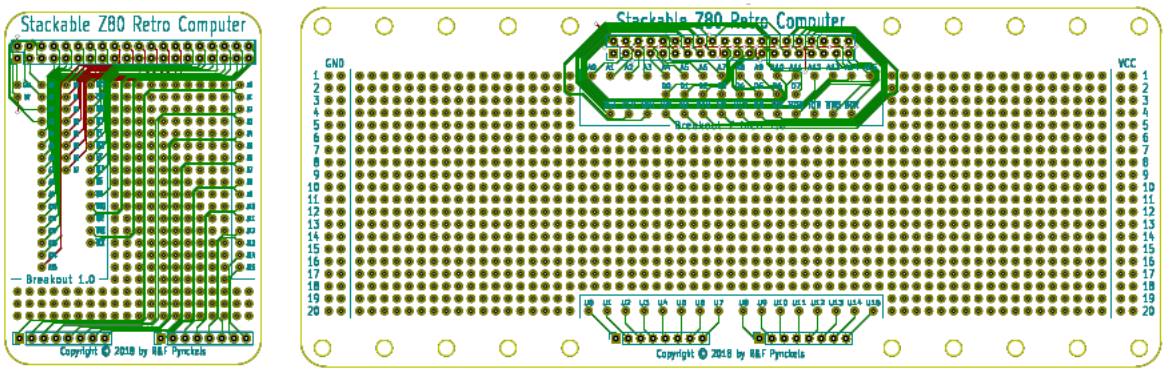

FIGURE 27 - BREAKOUT PCB DESIGN

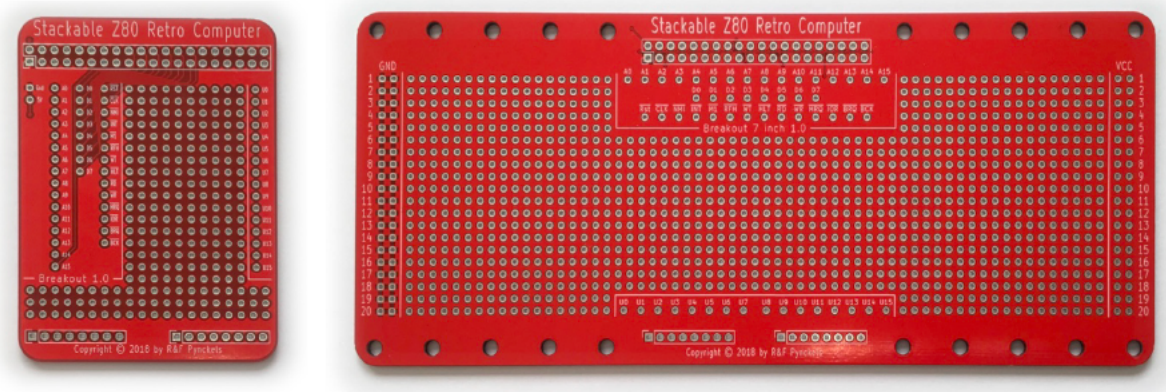

FIGURE 28 - BREAKOUT NON-POPULATED PCB'S

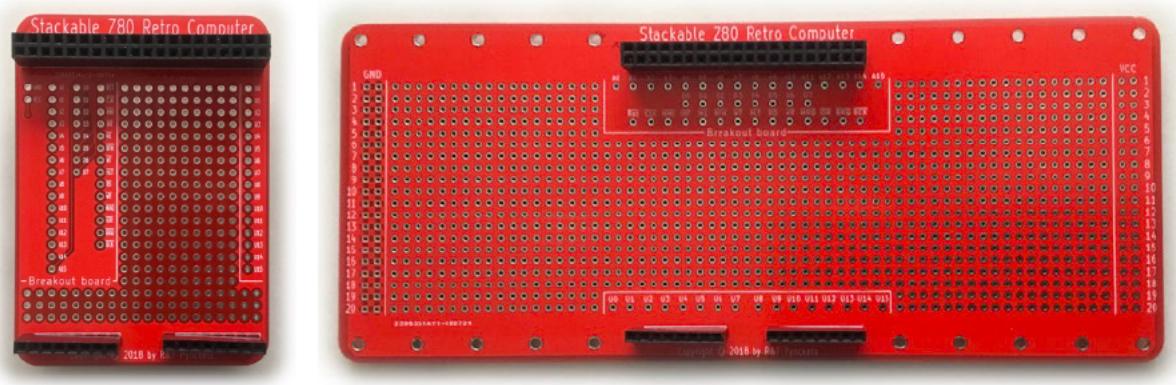

FIGURE 29 - BREAKOUT POPULATED PCB'S

8. Software development

8.1. Development environment

The development environment for the S80 consists of the following elements. Note that we only mention the software and hardware we have used, but that there are much more choices than the ones mentioned. We have chosen the environment elements that we feel comfortable with:

- Operating system: Linux, Mac OSx or Windows
- Terminal simulation: PuTTY, RealTerm, Terminal window (for Linux and Mac)
- Editor: Programmers Notepad, Atom Editor, Syntra Small, Xcode
- Cross-compiler: z88dk (z80asm, zcc, appmake)
- EEPROM programmer: MiniPro TL866 Universal Programmer

Note that a cross-compiler is a software package that runs on a certain platform (hardware/software) and compiles code to binary code that can be ran on another platform (hardware and/or software).

8.2. Developed software

During the development of the S80, a number of different small programs have been developed to facilitate the hardware testing:

- Serial test program (assembly)
- Memory test program (assembly)
- Microcode (a minimal BIOS that loads programs) and Microcode test program (assembly)
- General program: Towers of Hanoi (C program: test the S80 and the development environment)

In order to make life easier during development, we also developed a bash script (Linux and Mac) and/or batch file (Windows) for each program to automate the cross-compilation phase from source to EEPROM loadable file.

We will include a sample script (see *Source code 1*) and program (see *Source code 2*) below. However, in order not to overload this document and in order to accommodate the reader, we provide all source code (together with the functional and PCB schematics in a downloadable ZIP file (see the *bottom of the first page of this document* for a link).

The sample batch script (Windows) below (see *Source code 1*) belongs to the memory test software. The chosen file format is only dependent on the fact that we had a Windows machine available at the moment to test the S80 memory board.

SOURCE CODE 1 - MAKE.BAT

```
1.  @echo off
2.
3.  del *.bin 2>NUL
4.  del *.hex 2>NUL
```

```
del *.lis 2>NUL
del *.map 2>NUL
del *.o   2>NUL

z80asm   cpu=z80   list   make bin   map %1.asm
if %errorlevel% neq 0 exit %errorlevel%

del *.o   2>NUL

appmake +glue   binfile %1   clean   ihex
if %errorlevel% neq 0 exit %errorlevel%

ren *.ihx *.hex

del *.bin 2>NUL
del *.map 2>NUL
```

The assembly program below (see *Source code 2*) is the memory test software. The file is assembled with the above .bat (see *Source code 1*) to a .hex file. The .hex file is written to an EEPROM chip and plugged into the S80 memory board. When starting up or resetting the S80, the program starts running at address 0x0000. If all goes well, reading the address bus of the S80 with a protocol analyzer will show that the Z80 processor on the processor board reads instruction by instruction and jumps at the right moment from *Code_Low* to *Code_High* and back. If things go less will, you will see soon enough that the protocol analyzer doesn't give repetitive signals but just runs through the entire memory instead.

SOURCE CODE 2 - S80_MEMORY_TEST.ASM

```
;================================================================================
;
; Project: S80 memory test
; File:    S80_memory_test.asm
; Version: 1.0
;
; Created: 2018 10 20
; Adapted: 2018 10 21
;
; Author : Filip Pynckels
;
; Build tools (z88dk):
;
;       z88dk/z80asm
;       z88dk/appmake
;
; Build commands (Dos/Windows):
;
;       @echo off
;
;       del *.bin 2>NUL
;       del *.hex 2>NUL
;       del *.lis 2>NUL
;       del *.map 2>NUL
;       del *.o   2>NUL
;
;       z80asm   cpu=z80   list   make bin   map %1.asm
;       if %errorlevel% neq 0 exit %errorlevel%
;
;       del *.o   2>NUL
;
;       appmake +glue   binfile %1   clean   ihex
;       if %errorlevel% neq 0 exit %errorlevel%
;
;       ren *.ihx *.hex
```

```
;
;       del *.bin 2>NUL
;       del *.map 2>NUL
;
;===============================================================================
;
; This program is free software; you can redistribute it and/or modify it under
; the terms of the MIT Public License.
;
; This software is distributed in the hope that it will be useful, but without
; any warranty. Without even the implied warranty of merchantability or fitness
; for a particular purpose.
;
;===============================================================================


;===============================================================================
;=   H E A D E R   F I L E S                                                   =
;===============================================================================


;===============================================================================
;=   C O N S T A N T S                                                         =
;===============================================================================


;===============================================================================
;=   C O D E   S E G M E N T                                                   =
;===============================================================================

            org     0x0000

            ; Address 0x0000 code

Code_Low:
            nop
            nop
            nop
            jp      Code_High

            ; Fill with 0xFF bytes

            defs    0xFA, 0xFF

            ; Address 0x0100 code

Code_High:
            nop
            nop
            nop
            jp      Code_Low


;===============================================================================
;=   D A T A   S E G M E N T                                                   =
;===============================================================================

            .end
```

Conclusion

After developing the above hardware, and running the Towers of Hanoi program, sufficient evidence is given that the S80 is a functioning computer.

A number of next steps are considered:

- Design a compact processor board that contains the clock/power/reset functionality. This would permit to make a more compact S80 version 2.0 by dropping the two breakout boards, and by dropping the clock/power/reset board (hardware)
- Design a compact serial board. This would permit to make a more compact S80 version 2.0 when combined with the more compact processor board (hardware)
- Design a compact memory board. This would permit to make a more compact S80 version 2.0 when combined with the more compact processor and serial boards (hardware)
- Design the memory board to permit memory bank switching. This would permit to write programs that use up to 1Mb of memory (hardware)
- Design an Internet of Things board. For instance: a board that can communicate serial data through a laser transmitter/receiver with another S80 containing the same board (hardware)
- Extend the microcode to a full fledged BIOS containing the memory bank switching functionality and some useful I/O functions (software)
- Choose and write some other programs to demonstrate that the S80 is an effective computer (software)

Below, a number of appendices will be added shortly. The goal is to extend this document with extra information on the functioning of the S80, on new versions of the processor, serial and memory boards, on methods the reader can use to design, build, debug and program his own S80 boards and functionality, etc.

The appendix on debugging hardware design problems will be relatively extensive with emphasis on signal measuring, logic analysis, VHDL simulation, emulation of parts of the hardware (breadboarding) and emulation of the entire hardware (wire wrapping).

Appendix 1 - Processor board 2.0

A1.1. Processor 2.0 - Description

The first version of the processor board did not contain an own oscillator (CPU clock signal). On the other hand, it did contain protection circuitry against floating bus lines and against voltage peaks.

The first version of the S80 also contained a relatively large clock/power/reset board that, with the benefit of hindsight, was overshoot concerning the number of oscillators that could get installed.

Version 2.0 of the processor board should be smaller than version 1.0, it should not contain the protection circuitry, but it should contain the clock/power/reset functionality. This change makes the board smaller and more stand-alone. In fact, the only thing missing to have a functioning Internet of Things (IoT) computer would be some ROM and eventually some RAM memory. For IoT, all I/O can be sent to the user bus or via either bus to another S80 board.

Since the input connector also contains the Rx and Tx signals, besides the GND and VCC signals, the processor board 2.0 must pass the Rx and Tx signals via the user bus to the serial board. In order to be compatible with serial board 2.0, the Rx signal must be passed through U1 and the Tx signal must be passed through U9. To be compatible with serial board 1.0, the Rx signal must be passed through U1 and the Tx signal must be passed through U2, so a jumper cable must patch user bus U9 to user bus U3.

A1.2. Processor 2.0 - Technical specifications

Table 6 gives an overview of most of the information needed to integrate the processor board in the S80 stack.

Author/Creator	Robin & Filip Pynckels
License	MIT License (open hardware and software)
Remark	This board is tested with Serial board 1.0 and Memory board 1.0
Status	Production boards are ready. Do it yourself kits are looked into.
PCB Manufacturing	JLCPCB batch numbers: 2396351A-Y18-190119
Dimensions (W x H x D)	53.3mm (2.1") x 73.7mm (2.9") x 1.6mm (0.063")
Outgoing bus signals	A[0..15] (not buffered, can go in high impedance) C[4, 7..11] (not buffered, can go in high impedance) C5, C13 (not buffered, no high-impedance mode)
Incoming bus signals	C[0..3, 6, 12] (not clamped, accept high impedance mode) P[0, 1] (no protection build in)
Bidirectional bus signals	D[0..7] (not buffered)
User bus signals	U1 (Rx outgoing on U-bus & incoming on TTL conn pin 3) U9 (Tx incoming on U-bus & outgoing on TTL conn pin 4)
Reserved Memory addresses	-
Reserved IO addresses	-

TABLE 6 - PROCESSOR BOARD 2.0 - TECHNICAL SPECIFICATIONS

A1.3. Processor 2.0 - Functional schematics

To give an overview, we join a rescaled version of the functional schematics of the processor circuit in *Figure 30*.

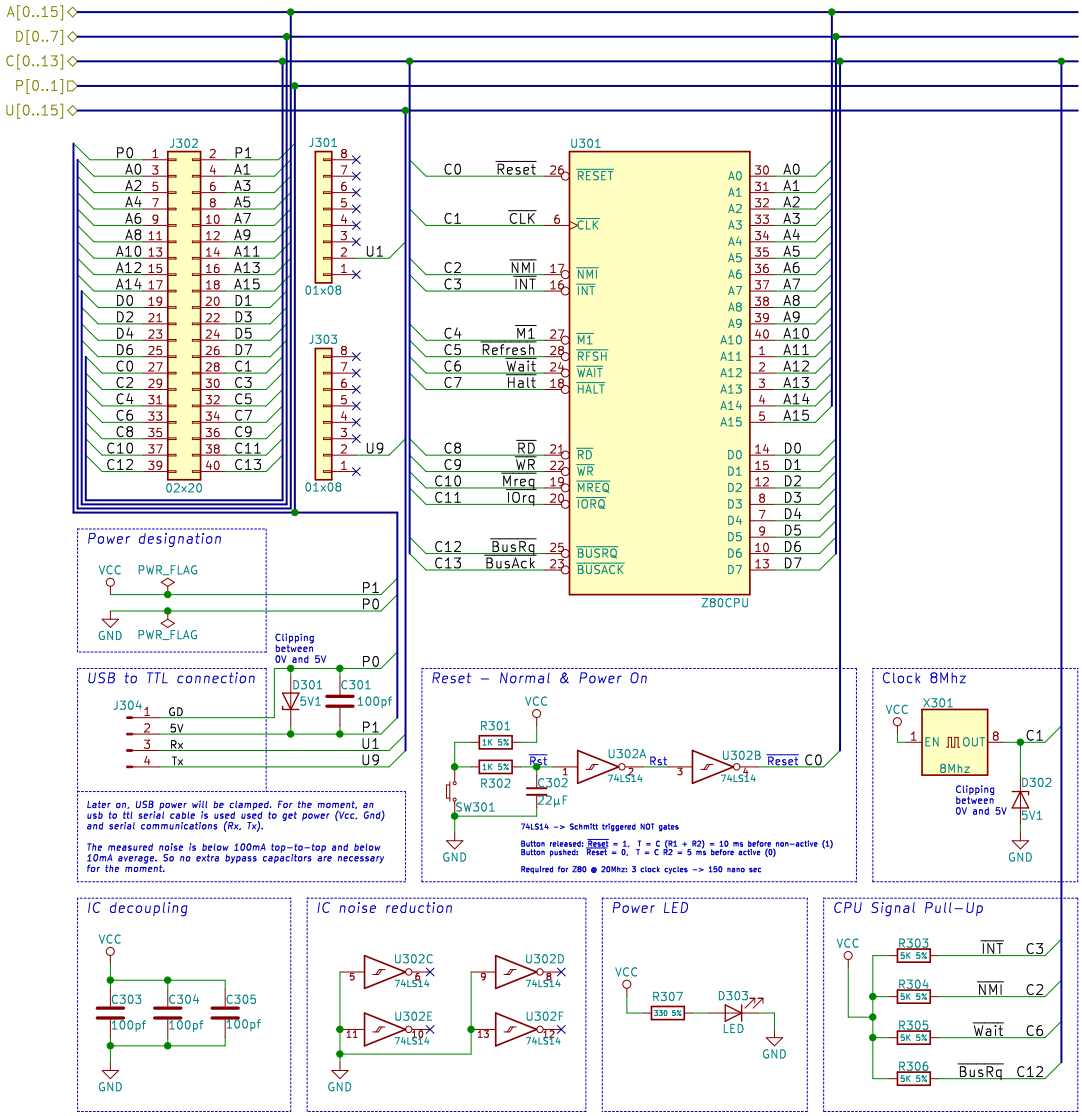

FIGURE 30 - PROCESSOR 2.0 FUNCTIONAL SCHEMATICS

A1.4. Processor 2.0 - PCB design

To give an overview, we join a rescaled version of the PCB design of the processor board in *Figure 31*. An impression of the non-populated and the populated final PCB is given in *Figure 32*. Please note that the below figures don't have the exact dimensions of the real PCB's

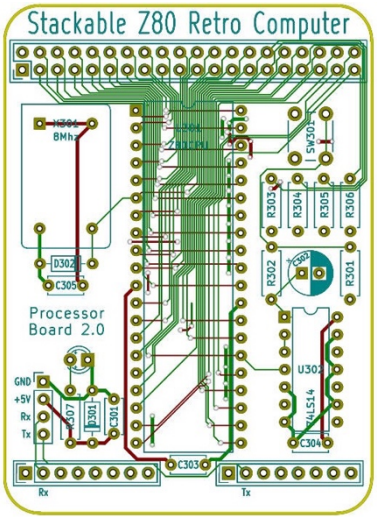

FIGURE 31 - PROCESSOR 2.0 PCB DESIGN

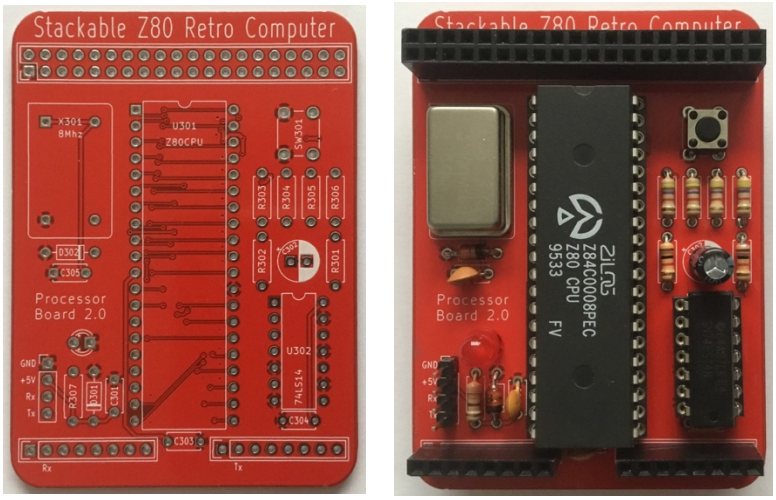

FIGURE 32 - PROCESSOR 2.0 PCB

A1.5. Processor 2.0 - Testing

We have done substantial testing to be sure that signals (in this case the reset signals) are crisp and that temperatures don't rise during an hour of intensive use of the board.

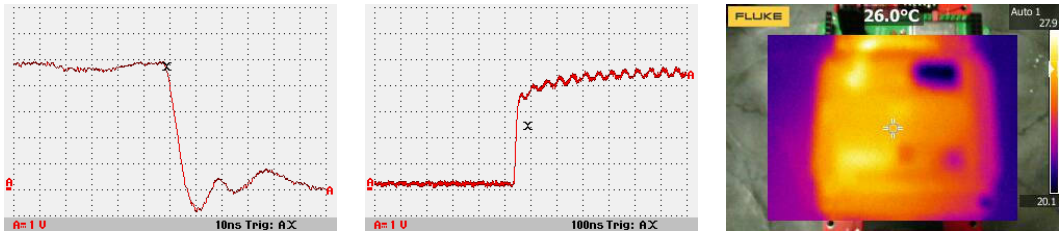

FIGURE 33 - PROCESSOR 2.0 TESTING

Appendix 2 - Serial board 2.0

A2.1. Serial 2.0 - Description

The first version of the processor board did not contain an own oscillator (CPU clock signal). On the other hand, it did contain protection circuitry against floating bus lines and against voltage peaks.

The first version of the S80 also contained a relatively large clock/power/reset board that, with the benefit of hindsight, was overshoot concerning the number of oscillators that could get installed.

Version 2.0 of the processor board should be smaller than version 1.0, it should not contain the protection circuitry, but it should contain the clock/power/reset functionality. This change makes the board smaller and more stand-alone. In fact, the only thing missing to have a functioning Internet of Things (IoT) computer would be some ROM and eventually some RAM memory. For IoT, all I/O can be sent to the user bus or via either bus to another S80 board.

Since the input connector also contains the Rx and Tx signals, besides the GND and VCC signals, the processor board 2.0 must pass the Rx and Tx signals via the user bus to the serial board. In order to be compatible with processor board 2.0, the Rx signal must be passed through U1 and the Tx signal must be passed through U9.

Multiple serial boards 2.0 can be used in the same "hardware stack", since serial board 2.0 permits to choose which user bus line is used for the Rx signal (U0...U7) and which user bus line is used for the Tx signal (U8…U15). So, if each serial board is used for as well Rx as Tx, 8 serial boards can be used together. If each serial board is used only for Rx or Tx, 16 serial boards can be used together (8 for Rx signal, 8 for Tx signals).

Please note that one serial board 1.0 and multiple serial boards 2.0 can be used together. Since serial board 1.0 uses U2 (Rx) and U3 (Tx), one must check carefully that neither of the serial boards 2.0 use the U2 nor the U3 for Rx signals, since this would come into conflict with the used serial board 1.0

A2.2. Serial 2.0 - Technical specifications

Table 7 gives an overview of most of the information needed to integrate the processor board in the S80 stack.

Author/Creator	Robin & Filip Pynckels
License	MIT License (open hardware and software)
Remark	The difference between the version 1 and version 2 board are on the one side its dimensions and on the other side the fact that a pin can be chosen on the User databus for the Rx and the Tx signal. This has as a consequence that multiple serial boards version 2 can be used with different I/O addresses.

	This board is tested with Processor board 2.0, Memory board 1.0, in combination with one Serial board 1.0 and in combination with one Serial board 1.0 and multiple Serial boards 2.0.
Status	Production boards are ready. Do it yourself kits are looked into.
PCB Manufacturing	JLCPCB batch numbers: 2396351A-Y19-190119
Dimensions (W x H x D)	76.2mm (3.0") x 73.7mm (2.9") x 1.6mm (0.063")
Outgoing bus signals	C3
Incoming bus signals	A[0..7] (checked on address range set by dipswitches) C0, C4, C8, C9, C11 (no protection build in) P[0, 1] (no protection build in)
Bidirectional bus signals	D[0..7] (buffered)
User bus signals	Rx on U[0..7] (outgoing on U-bus & TTL conn pin 3) Tx on U[8..15] (outgoing on U-bus & TTL conn pin 4)
Reserved Memory addresses	-
Reserved IO addresses	Variable base address in multiples of 0x08. Advised addresses are 0xC0 (base address) :: 0xC7

TABLE 7 - SERIAL BOARD 2.0 - TECHNICAL SPECIFICATIONS

A2.3. Serial 2.0 - Functional schematics

To give an overview, we join a rescaled version of the functional schematics of the processor circuit in *Figure 34*.

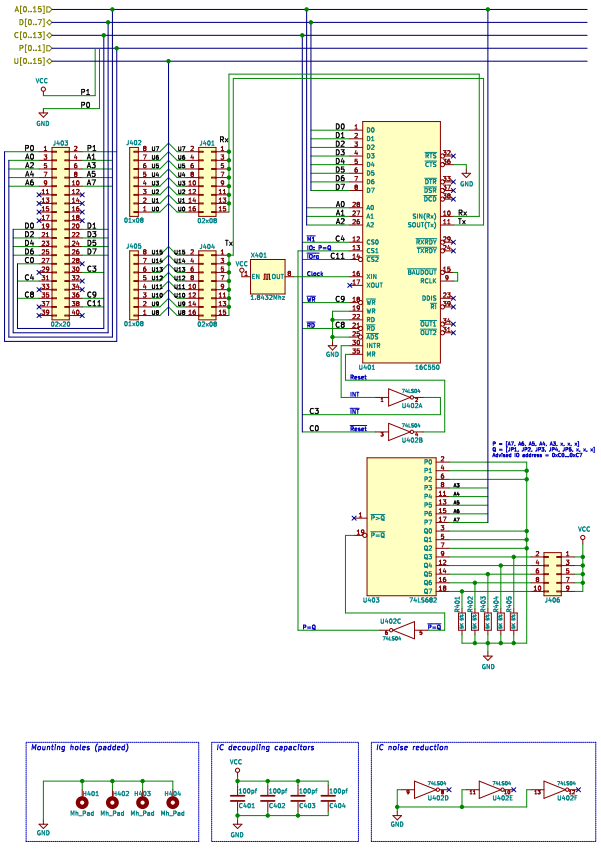

FIGURE 34 - SERIAL 2.0 FUNCTIONAL SCHEMATICS

A2.4. Serial 2.0 - PCB design

To give an overview, we join a rescaled version of the PCB design of the serial board in *Figure 35*. An impression of the non-populated and the populated final PCB is given in *Figure 36*. Please note that the below figures don't have the exact dimensions of the real PCB's

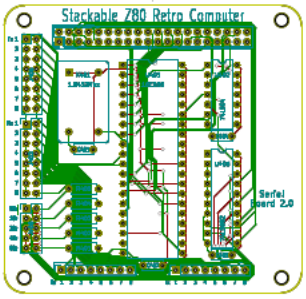

FIGURE 35 - SERIAL 2.0 PCB DESIGN

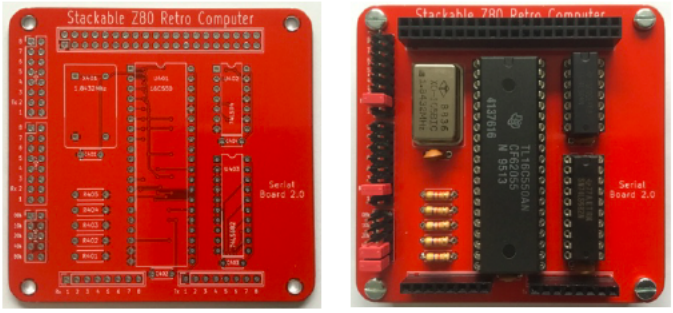

FIGURE 36 - SERIAL 2.0 PCB

For clarity: the upper 2x8 pin connectors with 1 jumper defines on which user bus line (U0..U7) the Rx signal is supposed to be. The lower 2x8 pin connectors with 1 jumper defines on which user bus line (U8..U15) the Tx signal is supposed to be.

To be compatible with the clock/power/reset board 1.0, the jumpers on both the 2x8 connectors should be set besides the label 2. For clarity concerning the backward compatibility, we have also put the labels Rx and Tx besides the label 2.

Below, the temperature reading of the serial 2.0 board after running the Towers of Hanoi program for 60 minutes.

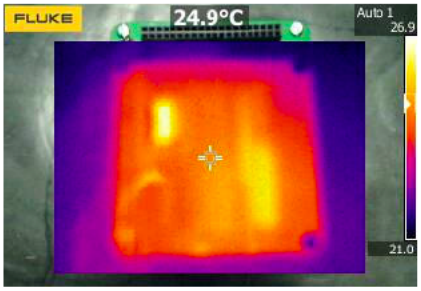

FIGURE 37 - SERIAL 2.0 TEST

Appendix 3 - Memory board 2.0 with bugs

A3.1. Memory board 2.0 – Find the bug…

The information below is erroneous! There are bugs to solve in the addressing logic of the memory board 2.0

The below information shows the actual state of play, to be an introduction to another document: ***2019-3-S80-retro-Z80-computer-with-stackable-segments_ver_2-0.pdf*** that will be published on the same website and go into the details of the redesign of the board presented below. Why? Because it doesn't function properly.

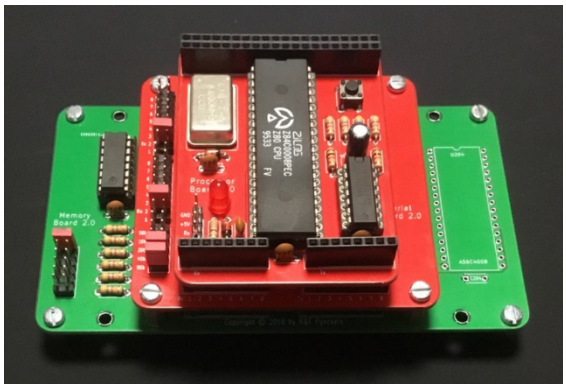

FIGURE 38 - MEMORY 2.0 WITH BUGS - TEST SETUP

The first action that has been taken is the simulation of the addressing logic. An example of a part of the VHDL component is shown below.

```
1  ----------------------------------------------------------------------------
2  --
3  -- Project: Library
4  -- File:    SN74LS682.vhd
5  -- Version: 1.0
6  --
7  -- Author : Filip Pynckels
8  --
9  -- Created: 2018 12 02
10 -- Adapted: 2018 12 03
11 --
12 -- Build & Simulation tools:
13 --
14 --      Quartus Prime 18.1.0
15 --      Modelsim
16 --
17 -- Functionality:
18 --
19 --      8 bit comparator with active low output for P=Q and P>Q
20 --
21 ----------------------------------------------------------------------------
22
23
24 ----------------------------------------------------------------------------
25 -- L I B R A R I E S
26 ----------------------------------------------------------------------------
27
28 library IEEE;
29
30     use IEEE.STD_LOGIC_1164.all;
31
32
33 ----------------------------------------------------------------------------
34 -- E N T I T Y   S N 7 4 L S 6 8 2
35 ----------------------------------------------------------------------------
36
37 entity SN74LS682 is
38
39     port(
40         P  : in STD_LOGIC_VECTOR(7 downto 0);                  -- Value 1
41
42         Q  : in STD_LOGIC_VECTOR(7 downto 0)                   -- Value 2
43             := ( others => '1' );                              -- Pull Up of 20kΩ
44
45         Not_P_Eq_Q  : out STD_LOGIC;                           -- Active low P=Q
46         Not_P_Gt_Q  : out STD_LOGIC                            -- Active low P>Q
47         );
48
49     end SN74LS682;
50
51
52 ----------------------------------------------------------------------------
53 -- A R C H I T E C T U R E   S N 7 4 L S 6 8 2
54 ----------------------------------------------------------------------------
55
56 architecture SN74LS682 of SN74LS682 is
57
58     begin
59
60         Not_P_Eq_Q <= '0' after 30 ns when P = Q else '1';     -- Active low P=Q
61         Not_P_Gt_Q <= '0' after 30 ns when P > Q else '1';     -- Active low P>Q
62
63     end SN74LS682;
64
```

SOURCE CODE 3 -VHDL (SN74LS682 COMPONENT)

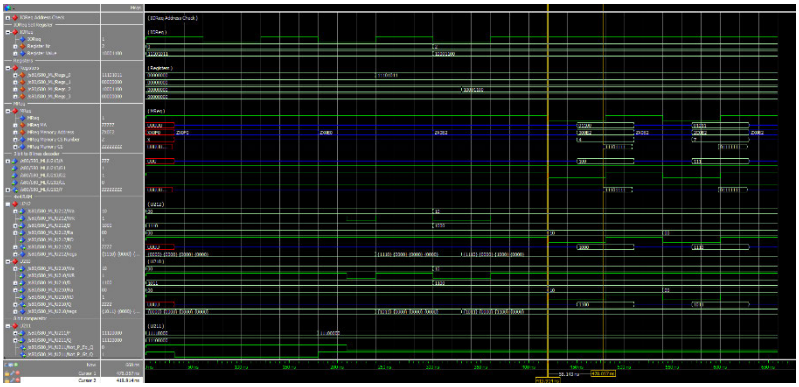

FIGURE 39 - VHDL SIMULATION RESULTS OF THE ADDRESSING LOGIC

Unfortunately, this first VHDL simulation shows us the results we needed and liked to see, without any error. This means that there is a problem with the addressing logic that is not simulated correctly. More specific, there is probably an initial or other behavior in the simulated components that is programmed as we interpreted that behavior instead of how it really functions.

So, some actions have to be taken to assess the real cause of the problem:

- Establish the exact behavior of the different logic IC's that are used. The VHDL simulation and the first build of the erroneous Memory 2.0 board start from our interpretation of the data sheet, but is that interpretation correct? An independent breadboard simulation of each components will be done, and the results will be compared to the respective datasheet (hardware).

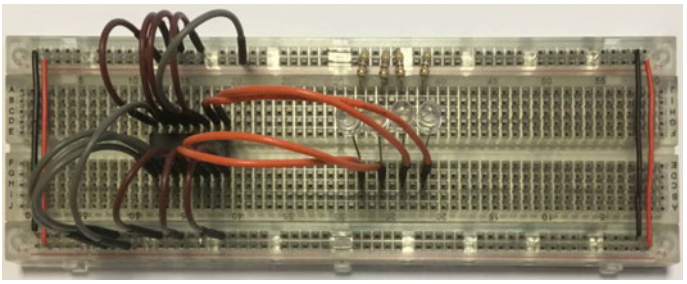

FIGURE 40 - BREADBOARD TEST OF THE SN74LS670 COMPONENT

- Establish the correlation of the behavior of all the logic IC's working together. A breadboard test setup will be constructed of the entire addressing logic. If this seems to function, a test-board (wire-wrapping) setup of the addressing logic with memory modules on a breadboard will be constructed to further assess the problem (hardware).

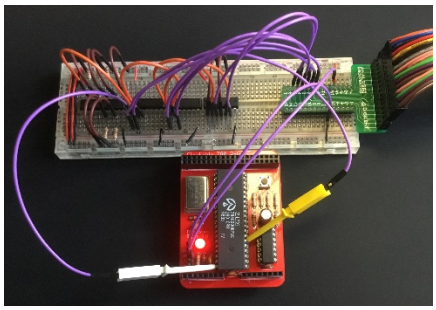

FIGURE 41 - BREADBOARD TEST OF THE MEMORY ADDRESSING LOGIC

- The erroneous Memory 2.0 board will be analyzed with a protocol analyzer to see if small corrections can be made on the board PCB to make it function anyway, or at least to see where we have gone so terribly wrong (hardware).

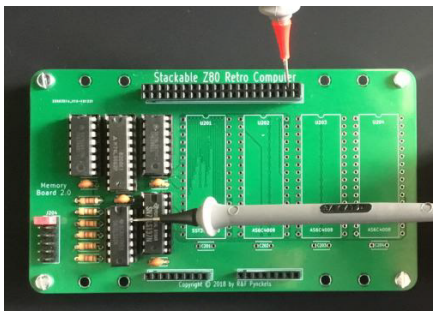

FIGURE 42 - ADDRESSING LOGIC DEBUGGING

The information below (*Figure 43* and in *Figure 44*) is used to do the tests as described above. This information is shown on an as-is basis as we already have established that there is at least one bug that prevents this schematic to be correct.

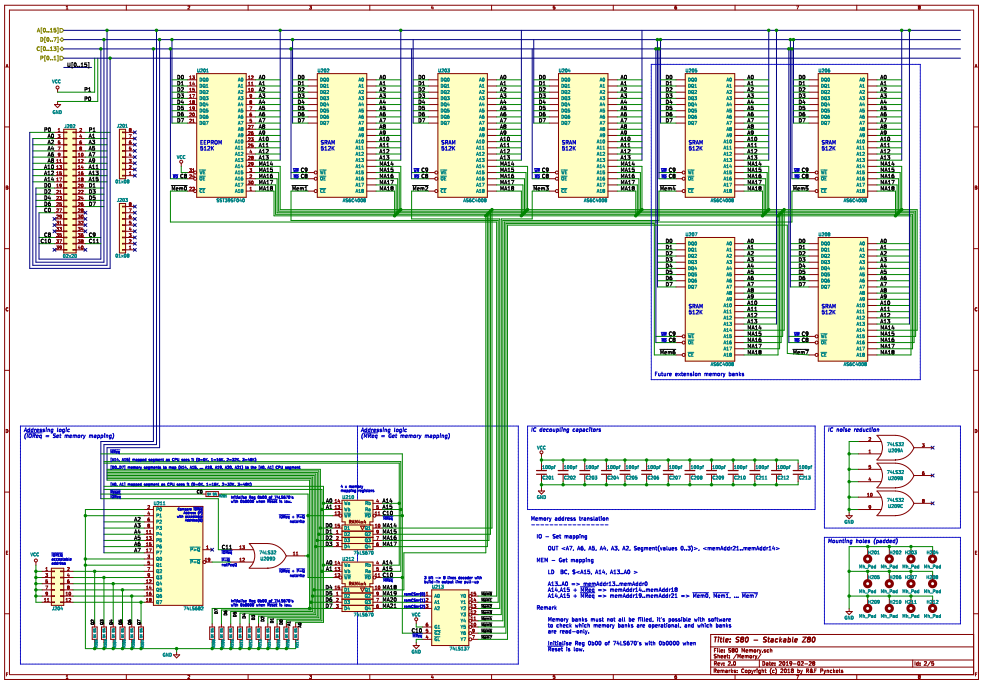

FIGURE 43 - MEMORY 2.0 WITH BUGS - FUNCTIONAL SCHEMATICS

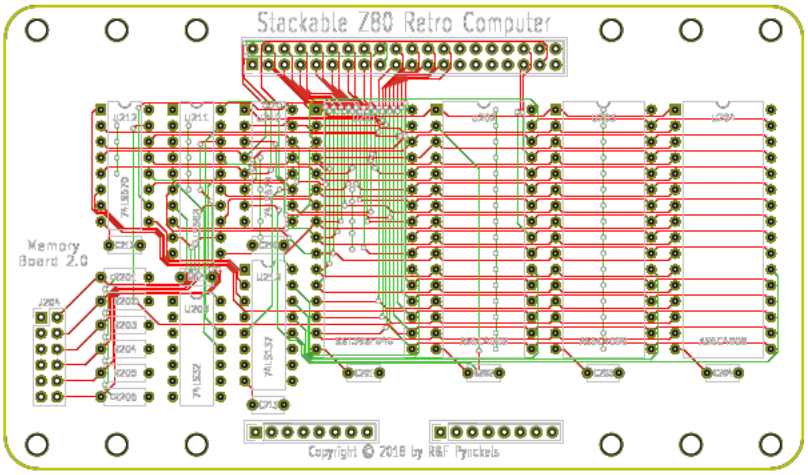

FIGURE 44 - MEMORY 2.0 WITH BUGS - PCB DESIGN

A3.2. Memory board 2.0 – Found the bug!

After some fiddling around, it became clear that the initialization of the SN74LS670 (the 4 x 4bit RAM) was different in theory (datasheet) than in practice. More specific, at initial power-up, the RAM registers seem to contain the value 0b1111 for the full 100% of the times we have done the tests (several 100ths of times).

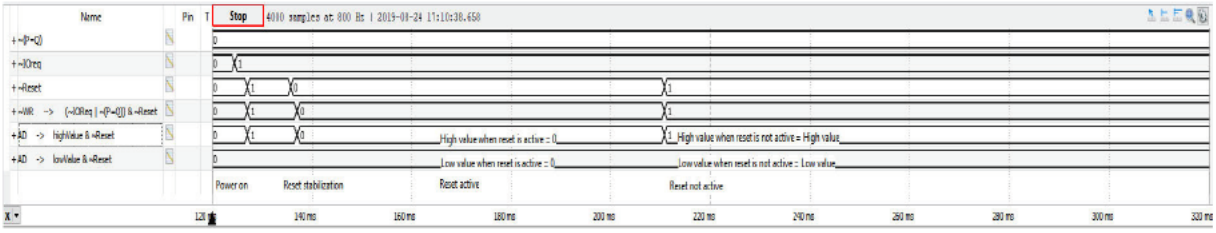

FIGURE 45 - ADDRESSING LOGIC ANALYSIS

However, we need the first RAM register to contain 0b0000 at startup, so the computers first 16K of memory addresses is mapped to the first 16K real memory addresses (more specific to the start of the EEPROM memory).

Since the real behavior of the SN74LS670 did not do what we thought it did, at the start of the Z80 CPU, the first instruction that was fetched was an instruction almost at the end of the real memory, or even an instruction that was looked for in an not present memory IC.

Theoretically: "*Problem solved*".
Practically: "*Oh god, why art thou hating me…*"

As already stated before: the correction of the design (you can call it the development of a new design), the testing of the new design and the creation of the corresponding PCB will be discussed in a next document on the same website:

2019-3-S80-retro-Z80-computer-with-stackable-segments_ver_2-0.pdf