Restoration of S.D.Systems Z80 Starter Kit

written by Sergio Gervasini for ESOCOP – The European Society for Computer Preservation
http://www.esocop.org

Table of Contents

License

References

S.D. Systems Z80 Starter Kit operation manual:
https://www.esocop.org/docs/SDS_Z80_Starter_kit.pdf

ZBUG Dump
https://www.esocop.org/docs/SDS-ZBUG.zip

Demo programs for S.D.Systems Z80 Starter Kit
https://www.esocop.org/docs/sds-esocop-prom.zip

'zasm' - z80 assembler:
https://k1.spdns.de/Develop/Projects/zasm/

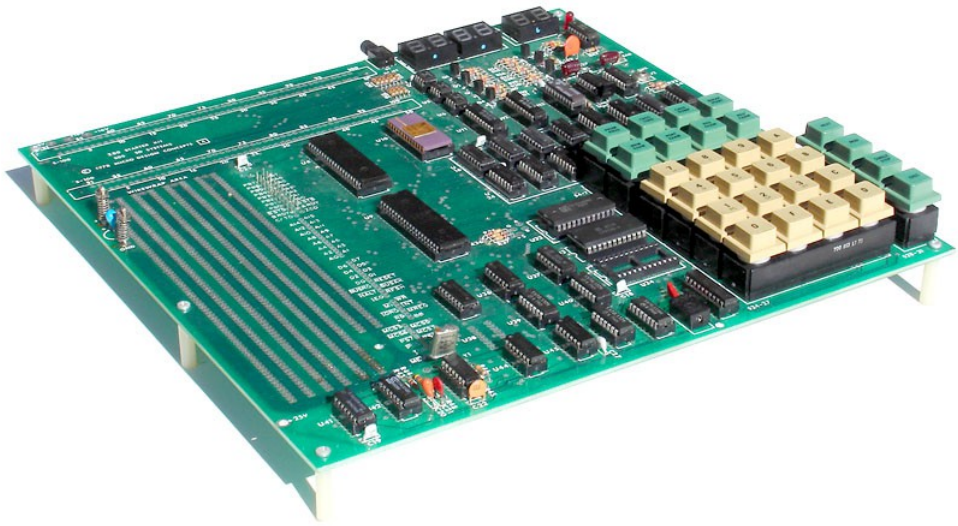
Z80-ASM
http://wwwhomes.uni-bielefeld.de/achim/z80-asm.html

Introduction

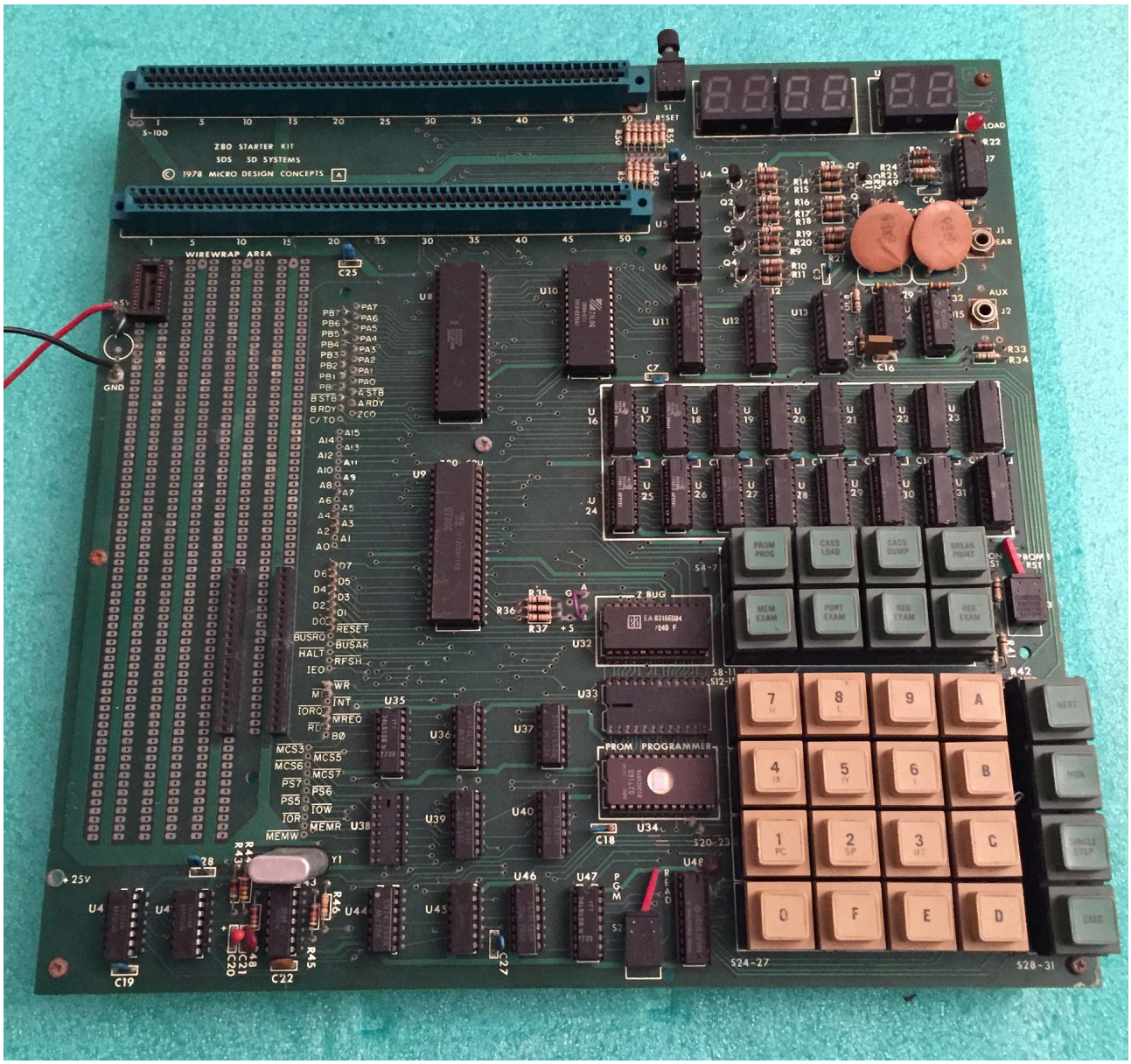
In the late 70's, the Z80 was considered the most powerful 8-bit processor available on the market, and its instruction set of 158 instruction types and clear, easy to learn mnemonics, probably made it the ideal processor on which to learn assembly language programming.

In 1979, S.D. Systems and Micro Design Concepts released the Z80 Starter Kit as a single board computer, with some unique features :
- a 2k program monitor called ZBUG with the capacity to: input code directly in memory through hexadecimal keybord and display, special keys useful to debug programs, read and write programs to audio cassette, program eprom directly on-board
- a Z80-PIO with 2 parallel interfaces
- a Z80-CTC with 4 independently programmable counter/timer channels
- 2 S100 bus connectors (although only 45 out of 100 signals are present)
- on-board wire wrap area

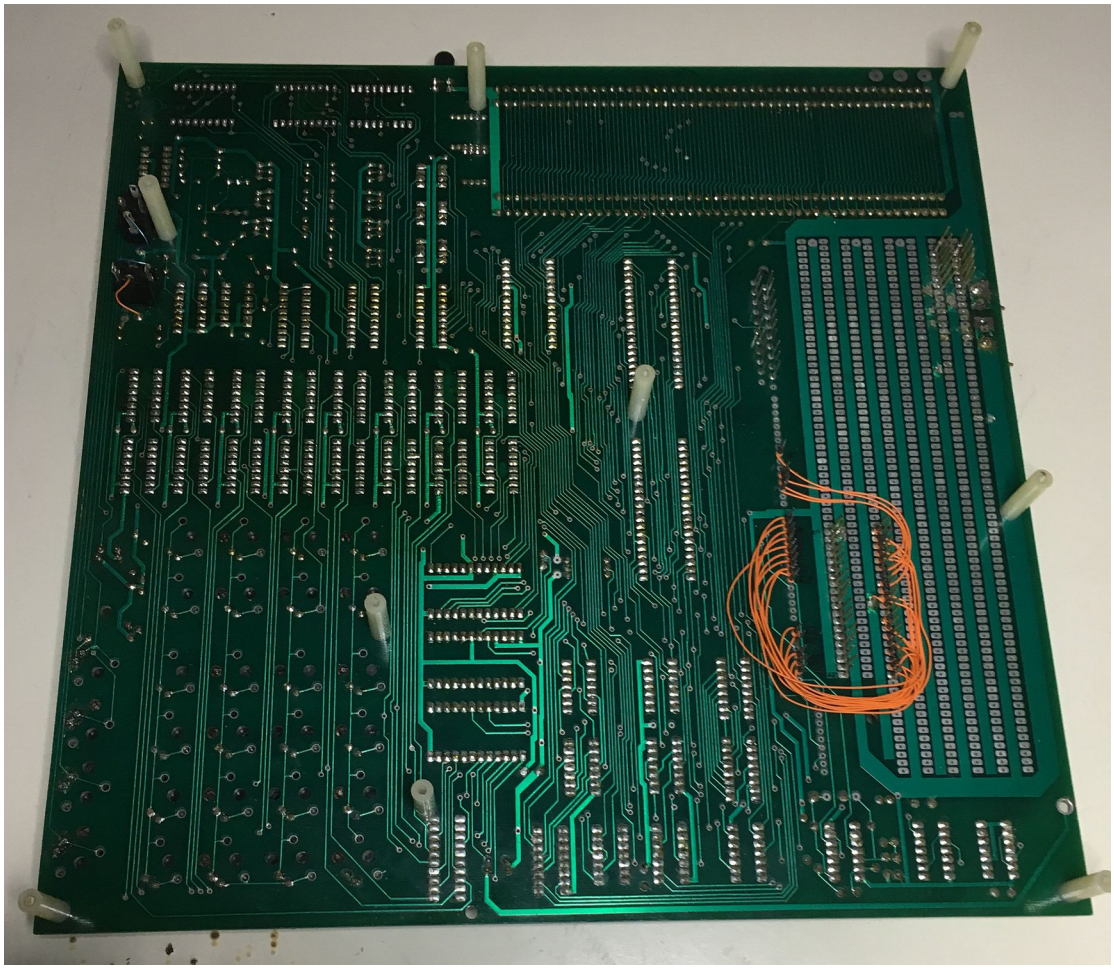
Able to run either 8080 or Z80 software, it was designed as the best value on the market for the hobbiest / experimenter / student who wants to learn about and work with microcomputers.



Our board



EsoCoP received this board from England apparently in fairly good condition with some sockets in the wire wrap area, a sign that previous owners used the card precisely for the purpose for which it was created.



However, a more careful analysis revealed several signs of bad state of conservation probably due to a humid environment.

Fixing

RAM socket

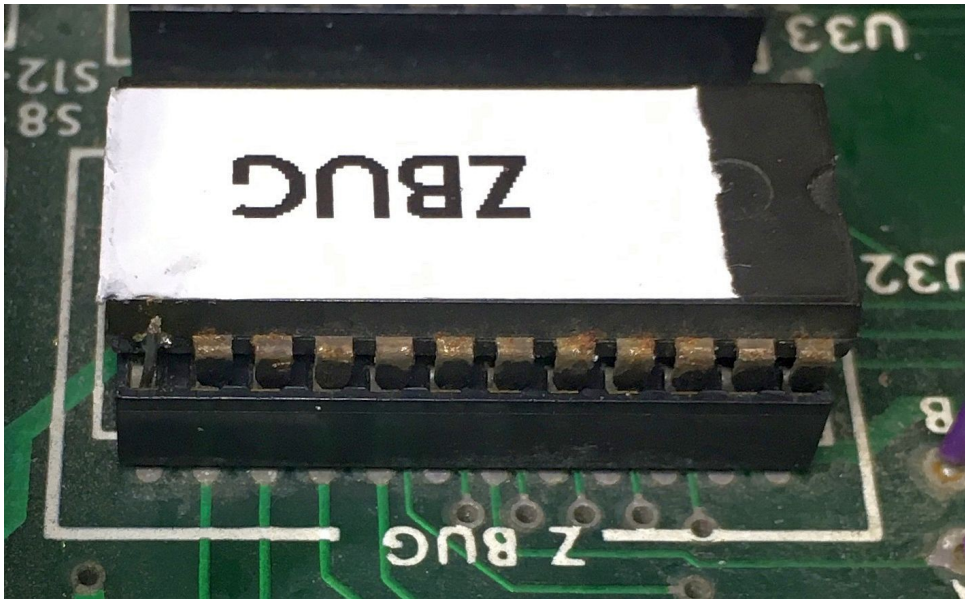
The card does not include power supply, so it was not necessary to do the usual tests before turning it on, the only measurement made was about the possible presence of short circuits.

We turned on the board through a laboratory DC power supply, but the result was null: display off and no data flow on the bus, the only existing signal: the clock.

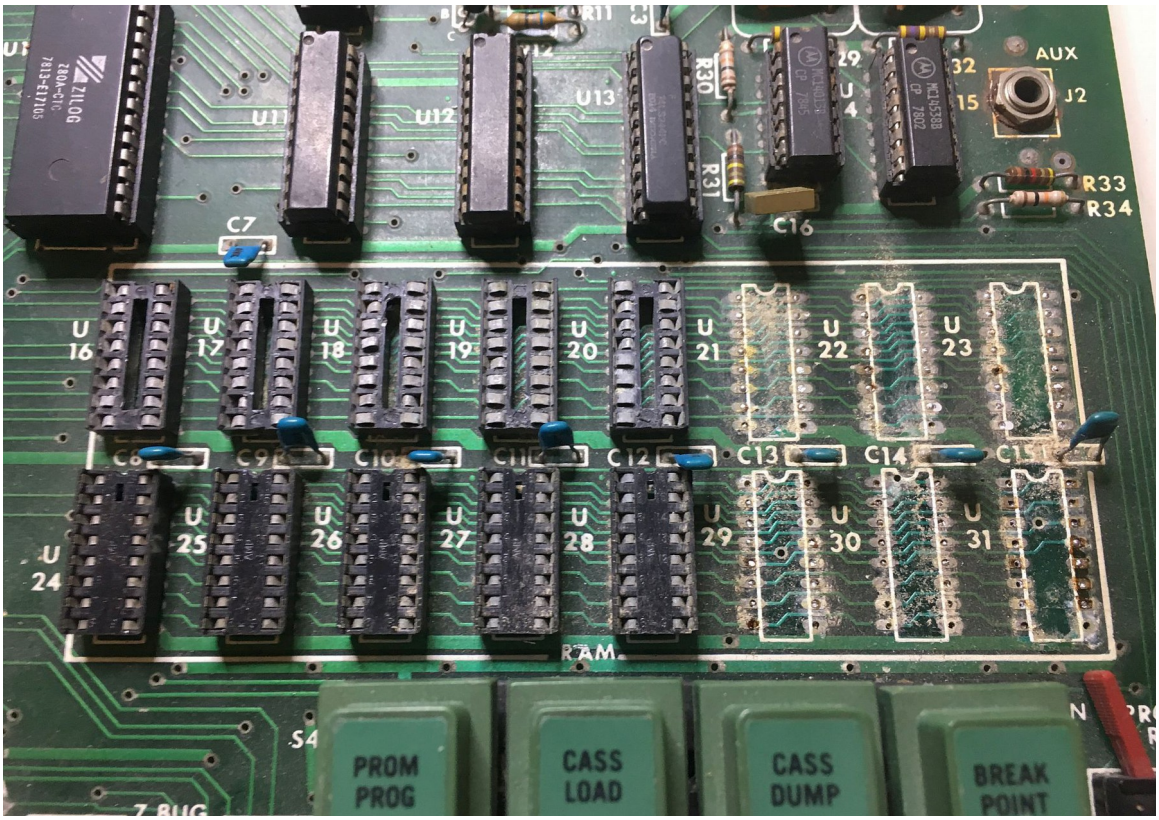
We decide to proceed in small steps, obviously the first one was to clean up, so we removed all the chips from the sockets cleaning carefully the pins and reinserting them.

Unfortunately, while we removed the ROM containing the ZBUG, one of the pins, very corroded, broke. Therefore it was necessary a delicate welding to solve this issue.

Before welding, as there was an high risk to damage the component, we took the opportunity to dump the content, now available to everyone on our site (see References).



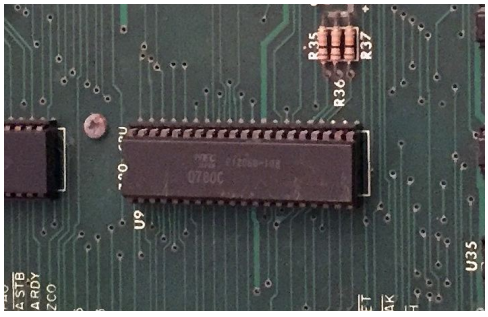
The sockets of the ram were corroded and damaged too (strangely they were a different model than the others sockets), so we decided to replace them, together with cleaning the board.



When we finisched to clean, replace the sockets, and check the connections between the CPU, RAM and ROM, there were no more contact problems.

CPU

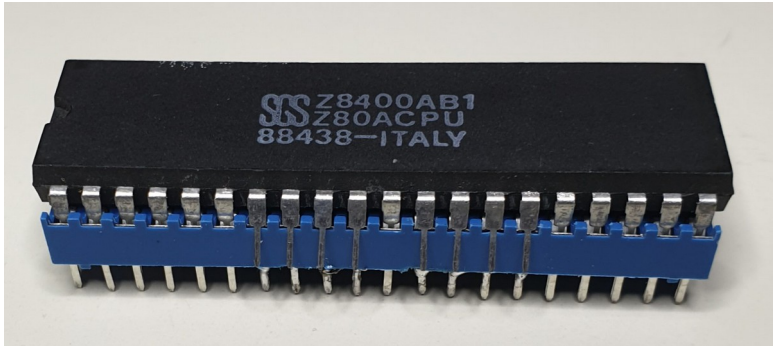
However a new attempt to turn on the board has not been more successful than the first one: always no data flow on the bus.



Making tests we realized that the CPU was warming more than normal, that's why we assumed a problem on that chip.

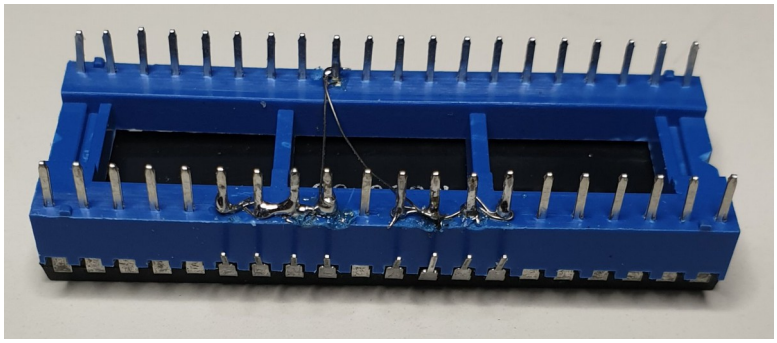
To better test it we made a small circuit based on a modified socket.

The idea was to ground all the data pins so that the CPU could carry out continuous NOP cycles and therefore be able to see the signals on the address bus.



Placing the CPU data pins raised outside the socket, we did test the CPU and parts of the board too.

Address bus data signals measurement let us know that the CPU was faulty; with a new CPU the signals were correct on the whole board.

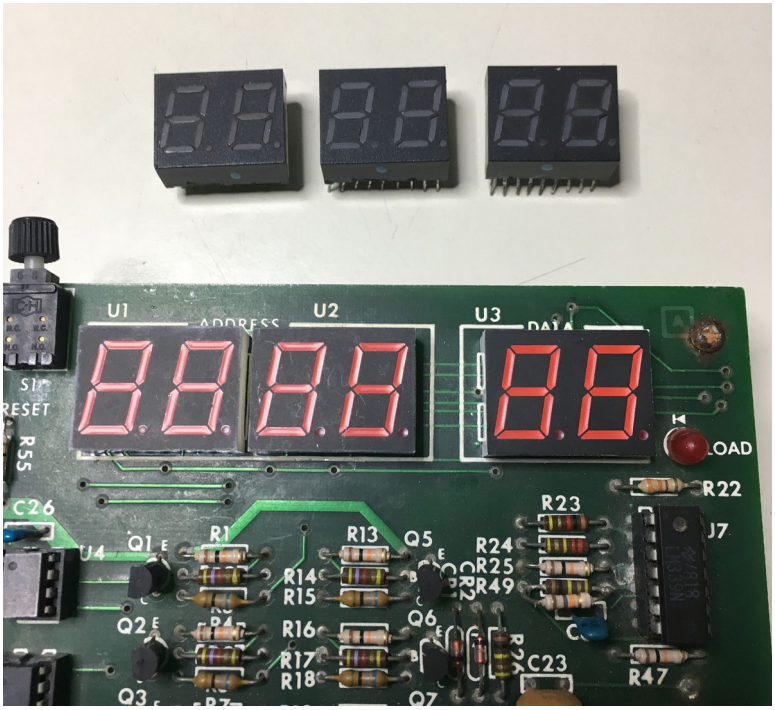


But ... even replacing the CPU with a new one, the displays didn't display anything!

Display

The next analysis has been made on the display driver circuits. Turning on one single segment at a time, made it possible to discover that most of the segments were faulty.

Today it is impossible to find those exact displays, but we have succeeded to find similar ones being able to replace them. After replacing all the displays with the new ones, tests on the visualization circuit finally gave positive results.

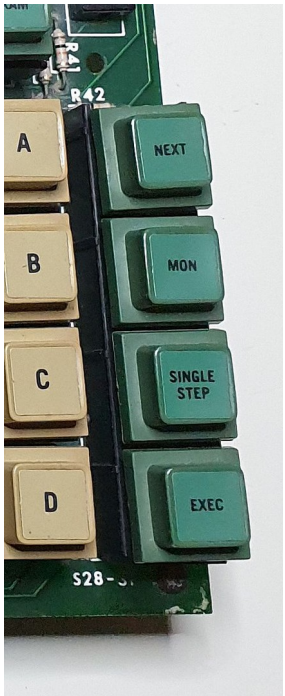


New power-on: We did press the reset and the display on the left turned on the central segment, which is the ZBUG prompt. But it just died in moments. Again.

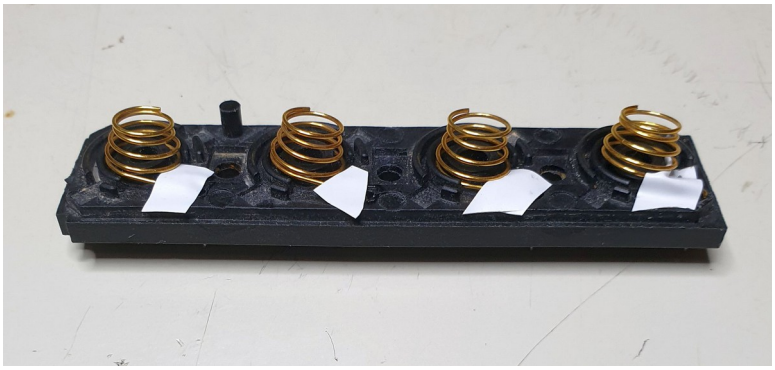
Keyboard

After a new complete measurement we discovered all the 4 green keys in the lower right corner short-circuited, as if they were always pressed, and this explained the brief appearance of the prompt.

The disassembly was quite easy. It allowed us to see the particular conformation of the keys, composed of two concentric springs of different diameters placed in contact by the buttons pressure.

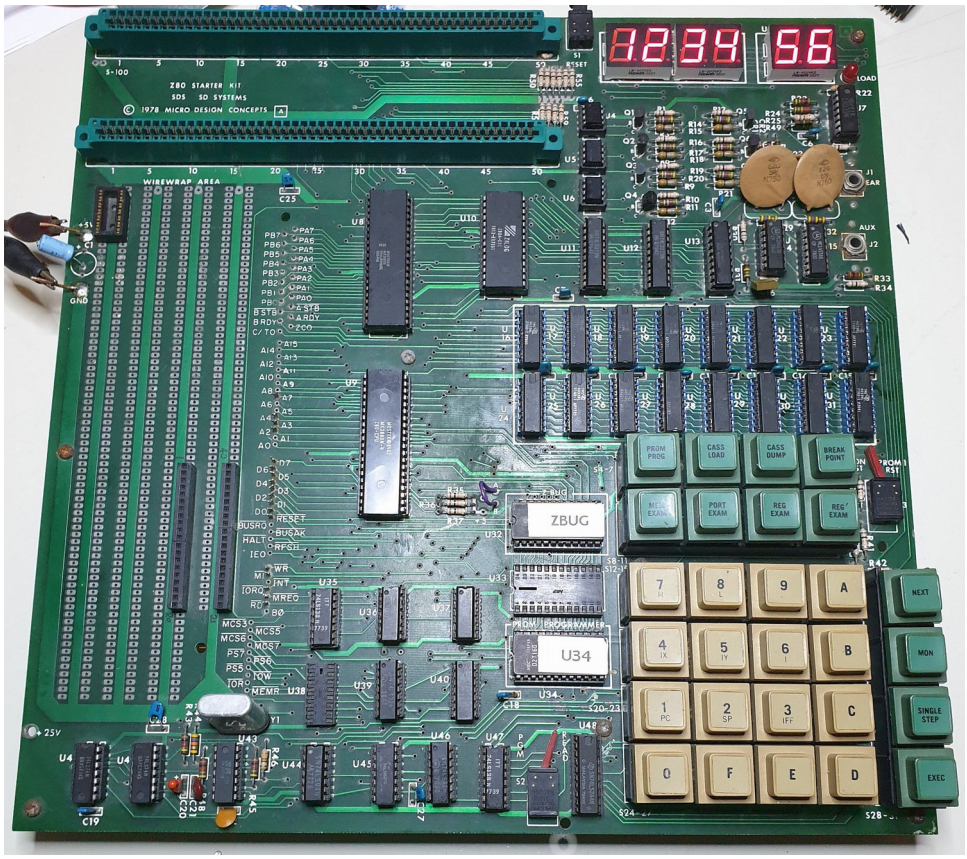


The short circuit problem was due to the base of the internal spring, interfering with the external one. To remedy we inserted a small piece of insulating tape to keep them detached.



Finally working!

When we put the keyboard back in place, we finally had the pleasure to see the stable prompt and we had been able to do some small tests using ZBUG.



Testing

Once we finished to work on the hardware, we tried some little programs: first we directly inserted opcodes with the keyboard and then we programmed some EPROMs. We did try 3 different programs.

Despite the years passed by, the Z80 assembly was still imprinted in our memories.

On the net we have been able to find some tools (see References) that helped us to compile and create content to put in an Eprom, that's why we did not use the programmer of the card, but another one connected to a modern PC.

Below you can find a small description of the three programs together with the assembly sources that have been compiled to binaries and burned in a 2716 Eprom, later placed in the U33 socket of the board.

Display test

In the manual (page 5-7) there is a small sample program to scroll the character 8 from right to left across the display.

It is quite simple to insert it directly via the keyboard, the effect is interesting and useful to verify that all display segments work properly.

Inspired by that program we created a custom one that is much more appealing, even if it does not involve all the segments.

Ram test

Making a ram test isn't really difficult, but we added two features: the content of the ram must not be modified by the test, it must test all the 256 combinations for each single byte.

Displaying the progress of the ram test created a bit more of complexity in the program and slowed down the execution, but with only 2k of memory this shouldn't be an huge problem.

Clock

With six 7-segments displays, the natural thing to do is to create a simple clock.

If you start this software from location 0x0900 the clock starts from 00:00:00. But, if you want to set a custom start time to the clock, just enter these simple instructions to location 0x2000 (beginning of user ram) and start from there.

```
2000: 06ss     ld b,ss        ; reg.B : ss=seconds
2002: 0Emm     ld c,mm        ; reg.C : mm=minutes
2004: 16hh     ld d,hh        ; reg.D : hh=hours (24h)
2006: C30609   jp 0x0906      ; start clock
```

The listings

```
                        ; ------------------------------------
                        ; zasm: assemble "sds-esocop-prom.asm"
                        ; date: 2019-04-15 18:35:37
                        ; ------------------------------------


                        ;
                        ; SOFTWARE FOR SD System Z80 Starter System
                        ; to be compiled and installed onto a 2716 eprom in U33
                        ;
                        ; (c) ESoCoP - Sergio Gervasini - Apr 2019
                        ;
                        ; sds-esocop-prom: 3 little programs to demonstrate
                        ; an SDS-Z80 system running
                        ;
                        ; can be compiled with zasm - z80 assembler
                        ; (c) 1994 - 2019 Günter Woigk.
                        ; homepage: https://k1.spdns.de/Develop/Projects/zasm/
                        ;
                        ;
                        ; ***** non destructive test memory *****
                        ; check memory with all pattern from 0 to 0xff
                        ; without deleting contents
                        ; if the test is successfull at the end
                        ; the display is 0000
                        ; otherwise it will display the addess location
                        ; and the pattern tested
                        ;

2000:                   STARTRAM    equ   0x2000
27FF:                   ENDRAM            equ   0x27ff
0088:                   OUTSEGM           equ   0x88
008C:                   OUTDIGIT    equ   0x8c
```

```
0800:                         ORG   0x0800

0800: 010020   [10]           ld bc,STARTRAM
0803: 11FF27   [20]           ld de,ENDRAM
0806: AF       [24]           xor a       ; zero acc.
0807: 60       [ 4]     loop1:     ld h,b
0808: 69       [ 8]           ld l,c
0809: 08       [12]           ex af,af'
080A: 7E       [19]           ld a,(hl)   ; get actual content and store in
                                          ; alternate reg.set
080B: 08       [23]           ex af,af'
080C: 77       [30]           ld (hl),a   ; put pattern test
080D: EDA1     [46]           cpi         ; check a with (hl)
                                          ; [note: hl++ and bc--]
080F: 201F     [53|58]        jr nz,error
0811: 2B       [59]           dec hl
0812: 08       [63]           ex af,af'
0813: 77       [70]           ld (hl),a   ; restore actual content
0814: 08       [74]           ex af,af'
0815: 23       [80]           inc hl
0816: 44       [84]           ld b,h            ; save hl
0817: 4D       [88]           ld c,l
0818: CD3E08   [105]          call showhl
081B: CD7808   [122]          call showa
081E: ED52     [137]          sbc hl,de   ; ram end?
0820: 20E5     [144|149]      jr nz,loop1
0822: 3C       [148]          inc a
0823: 2814     [155|160]      jr z,endok
0825: CD3E08   [172]          call showhl
0828: CD7808   [189]          call showa
082B: 010020   [199]          ld bc,STARTRAM
082E: 18D7     [211]          jr loop1

0830: 2B       [ 6]     error:     dec hl
0831: CD3E08   [23]           call showhl
0834: CD7808   [40]           call showa
0837: 18F7     [52]           jr error

0839: CD3E08   [17]     endok:     call showhl
083C: 18FB     [29]           jr endok

                        ; show the address in hl registers
083E: 08       [ 4]     showhl:    ex af,af'   ; save acc. in alternate reg.set
083F: 7D       [ 8]           ld a,l
0840: CD9D08   [25]           call outseg
0843: 3E04     [32]           ld a,4
0845: CD9508   [49]           call outdgt
0848: 7D       [53]           ld a,l
0849: CB3F     [61]           srl a       ; rotate acc. 4 times
084B: CB3F     [69]           srl a
084D: CB3F     [77]           srl a
084F: CB3F     [85]           srl a
0851: CD9D08   [102]          call outseg
0854: 3E08     [109]          ld a,8
0856: CD9508   [126]          call outdgt
0859: 7C       [130]          ld a,h
```

```
085A: CD9D08   [147]          call outseg
085D: 3E10     [154]          ld a,0x10
085F: CD9508   [171]          call outdgt
0862: 7C       [175]          ld a,h
0863: CB3F     [183]          srl a       ; rotate acc. 4 times
0865: CB3F     [191]          srl a
0867: CB3F     [199]          srl a
0869: CB3F     [207]          srl a
086B: CD9D08   [224]          call outseg
086E: 3E20     [231]          ld a,0x20
0870: CD9508   [248]          call outdgt
0873: CDB008   [265]          call testkey
0876: 08       [269]          ex af,af'   ; restore acc. from alt.register set
0877: C9       [279]          ret

                          ; display the content of the acc.
0878: F5       [11]     showa:     push af
0879: CD9D08   [28]           call outseg
087C: 3E01     [35]           ld a,1
087E: CD9508   [52]           call outdgt
0881: F1       [62]           pop af
0882: F5       [73]           push af
0883: CB3F     [81]           srl a
0885: CB3F     [89]           srl a
0887: CB3F     [97]           srl a
0889: CB3F     [105]          srl a
088B: CD9D08   [122]          call outseg
088E: 3E02     [129]          ld a,2
0890: CD9508   [146]          call outdgt
0893: F1       [156]          pop af
0894: C9       [166]          ret

                          ; pause for 1msec
0895: D38C     [11]     outdgt:    out (OUTDIGIT),a
0897: 3E2D     [18]           ld a,0x2d   ; about 1msec
0899: 3D       [ 4]     pause1:    dec a
089A: 20FD     [11|16]        jr nz, pause1
089C: C9       [21]           ret

                          ; take the 4 less significant bit of acc. and display
the content in 7-segments digit
089D: DD21C208 [14]     outseg:    ld ix, segm7
08A1: E60F     [ 7]     loops:     and 0x0f
08A3: 2805     [14|19]        jr z,show
08A5: 3D       [18]           dec a
08A6: DD23     [28]           inc ix
08A8: 18F7     [40]           jr loops
08AA: DD7E00   [19]     show: ld a,(ix)
08AD: D388     [30]           out (OUTSEGM),a
08AF: C9       [40]           ret

                          ; check for a key pressed, in this case the system
reload
08B0: 3E7F     [ 7]     testkey: ld a,0x7f
08B2: D388     [18]           out (OUTSEGM),a
08B4: 3E3F     [25]           ld a,0x3f
```

```
08B6: D38C     [36]          out (OUTDIGIT),a
08B8: DB90     [47]          in a,(0x90)
08BA: E61F     [54]          and 0x1f
08BC: FE1F     [61]          cp 0x1f
08BE: C20000   [71|71]       jp nz, 0
08C1: C9       [81]          ret

                        ; table bcd -> 7-segment display
08C2: C0F9A4B0          segm7:     defb
0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80,0x90,0x88,0x83,0xC6,0xA1,0x86,0x8E
08C6: 999282F8
08CA: 80908883
08CE: C6A1868E

                        ;
                        ; ***** clock *****
                        ; demo program to show a clock
                        ;
08D2: FFFFFFFF               ORG   0x0900
08D6: FF...

023A:                   DIVIDER    equ   0570  ; can be adjusted to
                                               ; obtain exactly 1 second delay

                        ; initialize registers
0900: 0600     [ 7]          ld b,0            ; reg.B : seconds
0902: 0E00     [14]          ld c,0            ; reg.C : minutes
0904: 1600     [21]          ld d,0            ; reg.D : hours (24h)
0906: 1E01     [28]          ld e,1            ; counter for digit
0908: 310023   [38]          ld sp,0x2300      ; init stack pointer
090B: 213A02   [48]          ld hl,DIVIDER

                        ; loop display refresh routine
090E: 78       [ 4]     showc0:    ld a,b            ; seconds
090F: F5       [11]     showc1:    push af
0910: CD9D08   [28]          call outseg
0913: 7B       [32]          ld a,e
0914: CD9508   [49]          call outdgt
0917: CB03     [57]          rlc e
0919: F1       [67]          pop af
091A: CB3F     [75]          srl a
091C: CB3F     [83]          srl a
091E: CB3F     [91]          srl a
0920: CB3F     [99]          srl a
0922: CD9D08   [116]         call outseg
0925: 7B       [120]         ld a,e
0926: CD9508   [137]         call outdgt
0929: CB03     [145]         rlc e
092B: 7B       [149]         ld a,e
092C: E6FB     [156]         and 0xfb    ; are we to the 3rd digit?
092E: 280C     [163|168]     jr z,showc
0930: 7B       [167]         ld a,e
0931: E6EF     [174]         and 0xef    ; are we to the 5th digit?
0933: 280A     [181|186]     jr z,showd
0935: CD4209   [198]         call intr   ; increment registers
0938: 1E01     [205]         ld e,1      ; reloop from the 1st digit
```

```
093A: 18D2     [217]          jr showc0
093C: 79       [ 4]     showc:     ld a,c
093D: 18D0     [16]           jr showc1
093F: 7A       [ 4]     showd:     ld a,d
0940: 18CD     [16]           jr showc1

0942: 2D       [ 4]     intr: dec l
0943: 2045     [11|16]        jr nz,endi
0945: 25       [15]           dec h       ; arrived to 1 sec?
0946: 2042     [22|27]        jr nz,endi
0948: 04       [26]           inc b       ; seconds +1
0949: 78       [30]           ld a,b
094A: E60F     [37]           and 0x0f
094C: FE0A     [44]           cp 0x0a     ; 10th seconds?
094E: 2037     [51|56]        jr nz,endx
0950: 78       [55]           ld a,b
0951: E6F0     [62]           and 0xf0
0953: C610     [69]           add a,0x10
0955: 47       [73]           ld b,a
0956: FE60     [80]           cp 0x60     ; 60th seconds
0958: 202D     [87|92]        jr nz,endx
095A: AF       [91]           xor a       ; zero a
095B: 47       [95]           ld b,a
095C: 0C       [99]           inc c
095D: 79       [103]          ld a,c
095E: E60F     [110]          and 0x0f
0960: FE0A     [117]          cp 0x0a     ; 10th minutes?
0962: 2023     [124|129]      jr nz,endx
0964: 79       [128]          ld a,c
0965: E6F0     [135]          and 0xf0
0967: C610     [142]          add a,0x10
0969: 4F       [146]          ld c,a
096A: FE60     [153]          cp 0x60     ; 60th minutes?
096C: 2019     [160|165]      jr nz,endx
096E: AF       [164]          xor a
096F: 4F       [168]          ld c,a
0970: 14       [172]          inc d
0971: 7A       [176]          ld a,d
0972: D624     [183]          sub 0x24    ; 24th hour?
0974: 280F     [190|195]      jr z,h24
0976: 7A       [194]          ld a,d
0977: E60F     [201]          and 0x0f
0979: FE0A     [208]          cp 0x0a     ; 10th hours?
097B: 200A     [215|220]      jr nz,endx
097D: 7A       [219]          ld a,d
097E: E6F0     [226]          and 0xf0
0980: C610     [233]          add a,0x10
0982: 57       [237]          ld d,a
0983: 1802     [249]          jr endx
0985: AF       [ 4]     h24:  xor a
0986: 57       [ 8]           ld d,a
0987: 213A02   [10]     endx: ld hl,DIVIDER
098A: C9       [10]     endi: ret
```

```
                        ;
                        ; ***** running segments *****
                        ; demo program to show a segment
                        ; at a time on the display
                        ; to simulate a circular motion
                        ;
098B: FFFFFFFF                ORG   0x0A00
098F: FF...
0A00: 061C      [ 7]    start:      ld b,028          ; lenght of data
0A02: 211B0A    [17]          ld hl,data
0A05: 7E        [ 7]    loop: ld a,(hl)
0A06: D388      [18]          out (OUTSEGM),a
0A08: 23        [24]          inc hl
0A09: 7E        [31]          ld a,(hl)
0A0A: D38C      [42]          out (OUTDIGIT),a
0A0C: 11FF30    [52]          ld de,0x30ff
0A0F: 1D        [ 4]    delay:      dec e
0A10: 20FD      [11|16]       jr nz,delay
0A12: 15        [15]          dec d
0A13: 20FA      [22|27]       jr nz,delay
0A15: 23        [28]          inc hl
0A16: 05        [32]          dec b
0A17: 20EC      [39|44]       jr nz,loop
0A19: 18E5      [51]          jr start
                        ; table segment, digit
0A1B: 7E20              data: defb 0x7e,0x20
0A1D: 7E10                    defb 0x7e,0x10
0A1F: 7E08                    defb 0x7e,0x08
0A21: 7E04                    defb 0x7e,0x04
0A23: 7E02                    defb 0x7e,0x02
0A25: 7E01                    defb 0x7e,0x01
0A27: 7D01                    defb 0x7d,0x01
0A29: 3F01                    defb 0x3f,0x01
0A2B: 3F02                    defb 0x3f,0x02
0A2D: 3F04                    defb 0x3f,0x04
0A2F: 3F08                    defb 0x3f,0x08
0A31: 3F10                    defb 0x3f,0x10
0A33: 3F20                    defb 0x3f,0x20
0A35: 6F20                    defb 0x6f,0x20
0A37: 7720                    defb 0x77,0x20
0A39: 7710                    defb 0x77,0x10
0A3B: 7708                    defb 0x77,0x08
0A3D: 7704                    defb 0x77,0x04
0A3F: 7702                    defb 0x77,0x02
0A41: 7701                    defb 0x77,0x01
0A43: 7B01                    defb 0x7b,0x01
0A45: 3F01                    defb 0x3f,0x01
0A47: 3F02                    defb 0x3f,0x02
0A49: 3F04                    defb 0x3f,0x04
0A4B: 3F08                    defb 0x3f,0x08
0A4D: 3F10                    defb 0x3f,0x10
0A4F: 3F20                    defb 0x3f,0x20
0A51: 5F20                    defb 0x5f,0x20

                              end
```