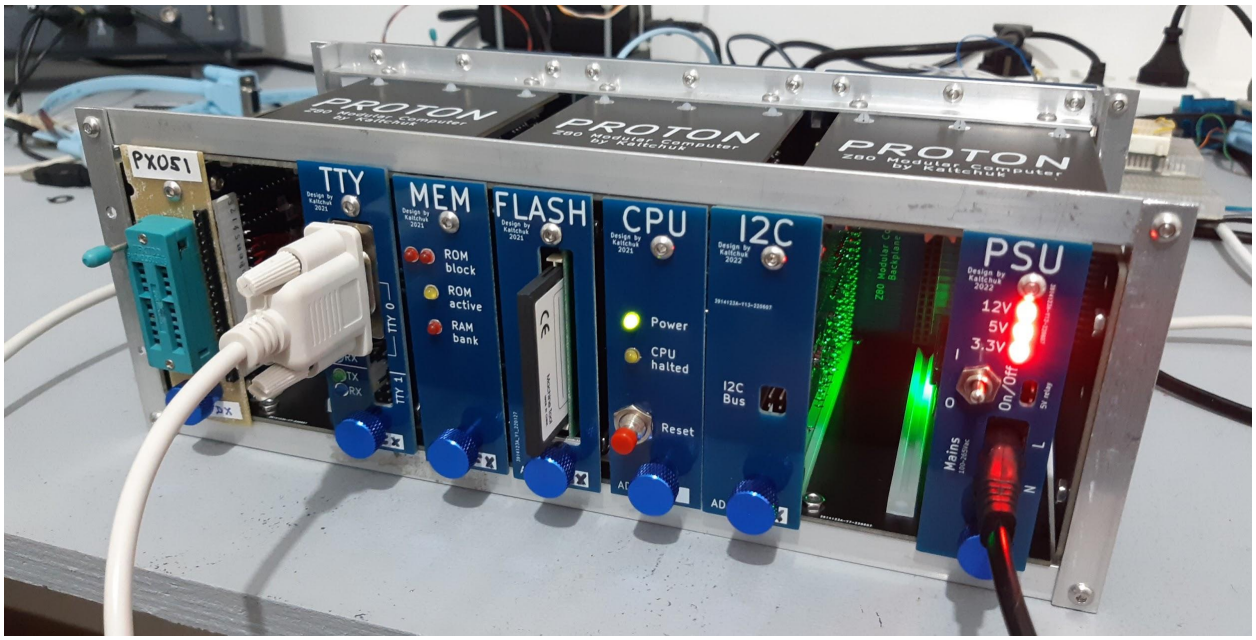
Proton - Z80 Modular Computer
User's Manual - version 1.0



Developed by Ricardo Kaltchuk
kaltchuk@gmail.com

Preface

Two years ago I was searching for some ICs in my component drawers and came across a Z80 CPU and an Intel 8251 USART. I had an instant flashback and remembered my first contact with a microprocessor back in 1982. It was a Z80. After a few minutes of nostalgia I decided to make good use of these components and build a computer.

Of course I couldn't avoid thinking about this "new project" with my Systems Engineer mindset. So, I established a few initial goals:

- This project is a hobby, so it has to be fun.
- It should use mainly technologies available in the 1980's.
- The computer has to run commercial software.
- The architecture should be modular allowing for new cards with specific functions.
- It also has to be a hardware/software development platform for students, hobbyists and enthusiasts.

After some research I decided to design the hardware fully compatible with CP/M 2.2.

One and a half years later I finally had my wire wrapped computer running CP/M. I could write programs using WordStar and compile them with a BDS C compiler or Microsoft Basic compiler. It was fun, exciting and challenging. After that, I decided it was time to turn the prototype into a finished product.

It's gratifying to see that in two years, alone, I was able to transform an idea into a final product.

Ricardo Kaltchuk
Ashdod, october 2021.

"Anyone who has brought up CP/M on a homebuilt computer has felt this moment of elation. A myriad of connections are properly closed; bits are flying at lightning speeds over buses and through circuits and program logic, to produce a single prompt".

Gary Kildall

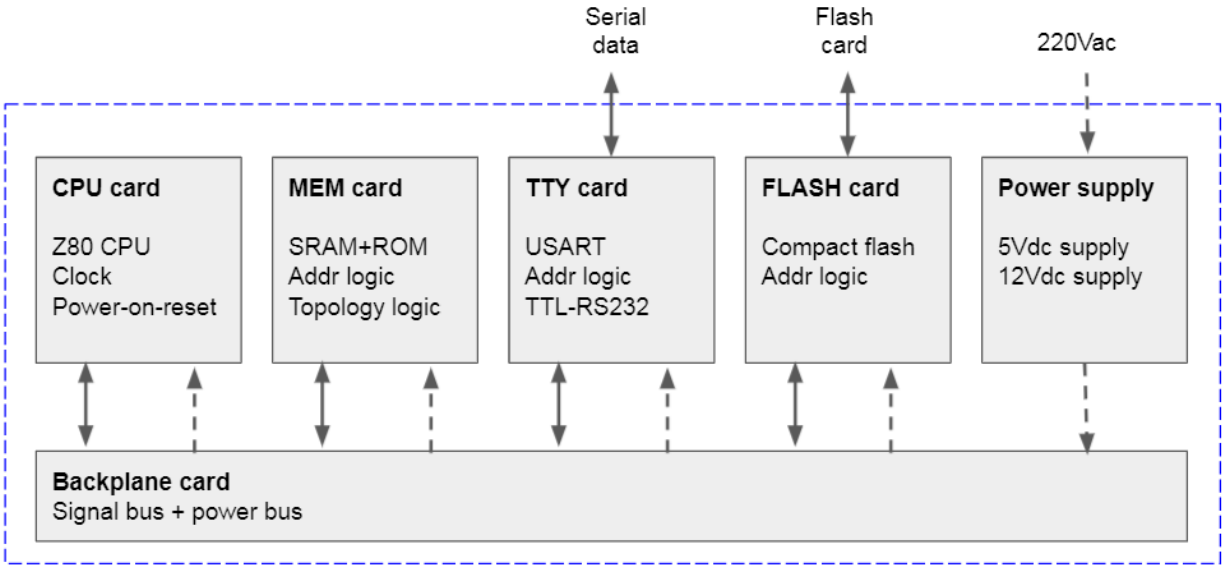
Index

Introduction

The Proton Z80 Modular Computer, as the name implies, is a modular computer based on the Z80 CPU. Is has a minimal configuration of five modules, plus the backplane card:

1. CPU Card - Z80 CPU, clock circuit and power-on-reset.
2. MEM Card - 64KB of ROM and 128KB of RAM.
3. FLASH card - 128MB of compact flash emulating 16 logical drives.
4. TTY Card - two serial ports, RS232 or TTL level.
5. PSU Card - 5V power supply.

Every card that exchanges data with the CPU, has a unique hexadecimal 4-bit I/O address ranging from 0 to Fh, that corresponds to the 4 most significant bits of the I/O address of the CPU. The addresses are configured via jumpers on each card. Even the MEM Card has an I/O address although it's not an I/O peripheral *per se* (more details on the MEM Card section). All the modules described above are assembled in a rack. The rack may contain more than one unit of each module, except for the CPU Card, MEM Card and PSU Card.



Proton Z80 Modular Computer system diagram.

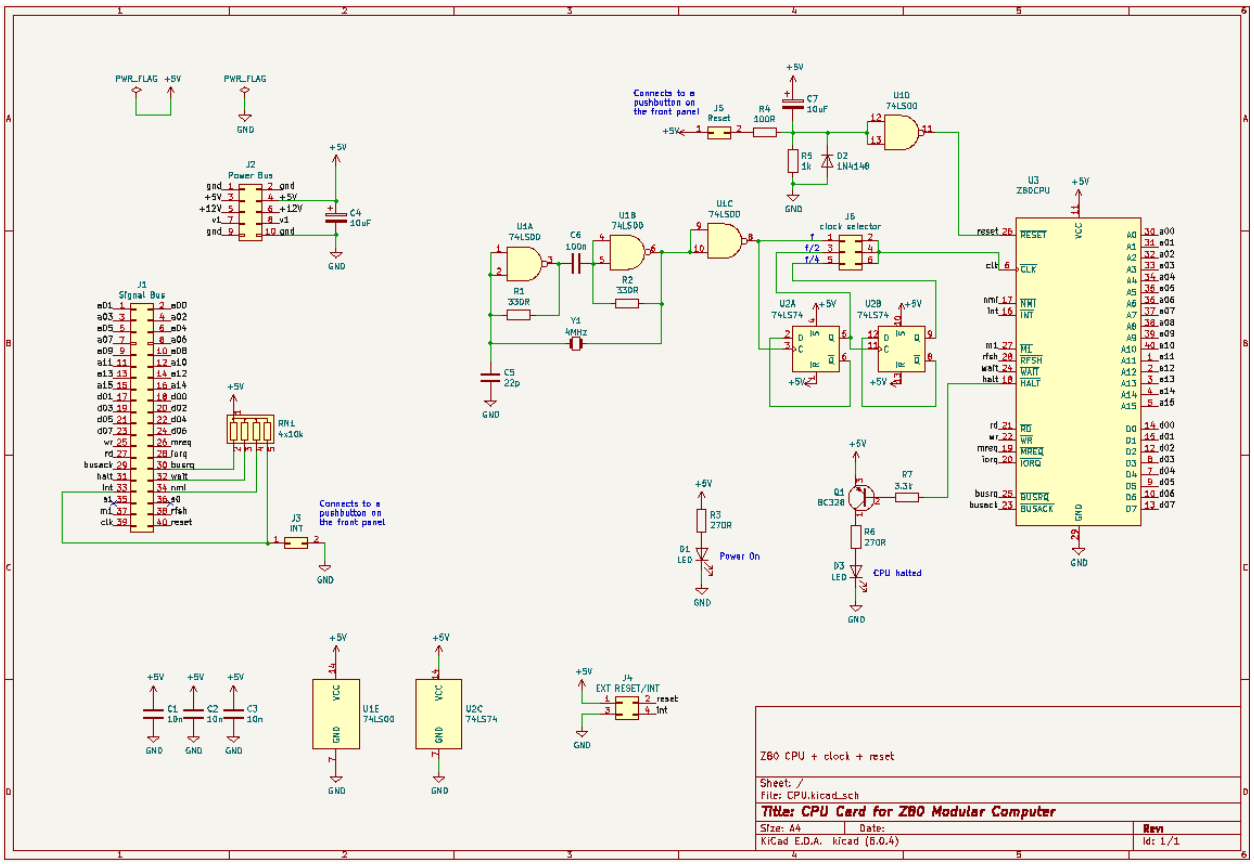
All the modules were first assembled and tested in wire wrapped protoboards and subsequently upgraded to a final PCB version. The FLASH Card was particularly tricky. On the wire wrapped prototype I only could manage to make the SanDisk compact flash work properly. The PCB version didn’t have this problem and even industrial compact flashes worked perfectly.

CPU Card

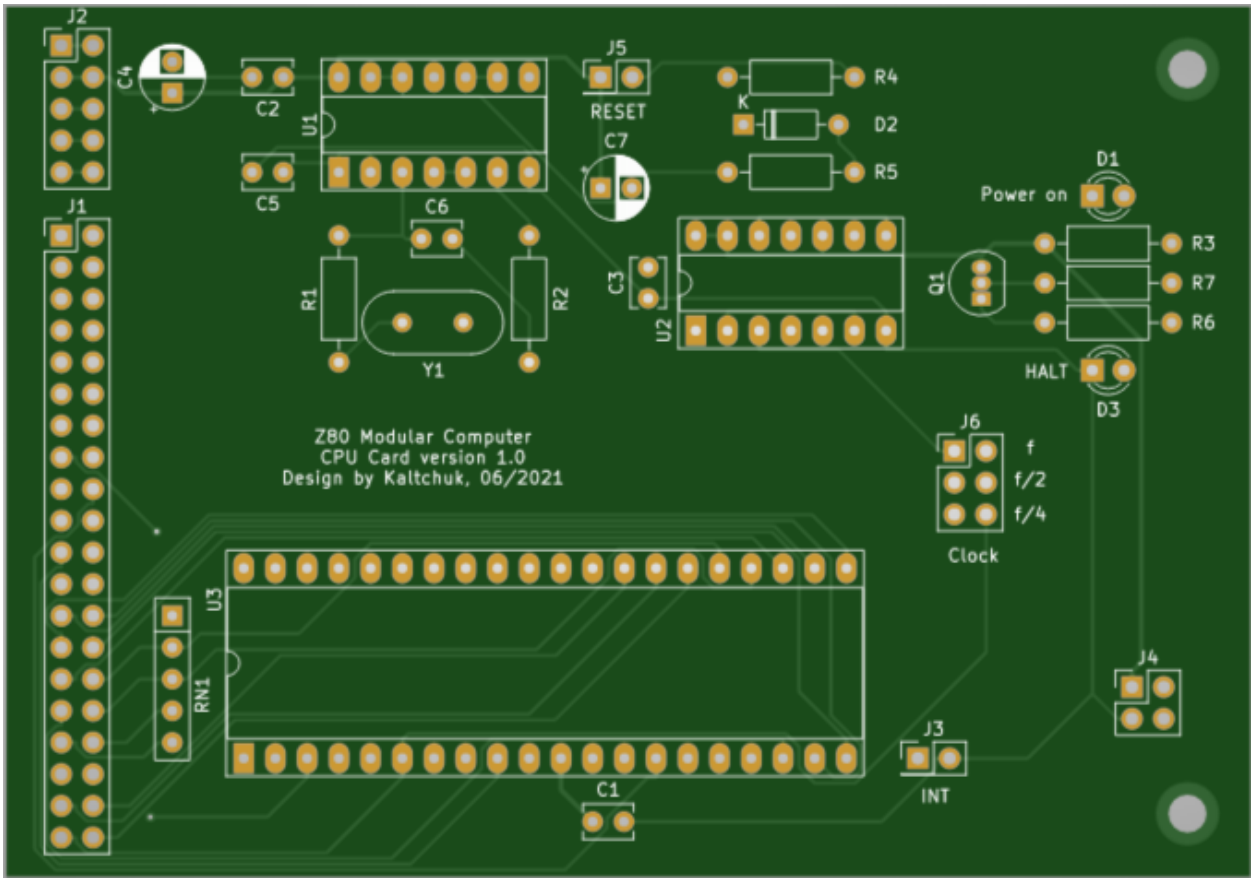
Description

The CPU Card has the Z80 CPU, the clock circuit and a power-on-reset circuit. The clock circuit uses an 8MHz crystal and two flip-flops to divide the frequency (4 and 2MHz).

The card frontal has a green LED to indicate power-on and a yellow LED that shows when the CPU is in HALT state. To reset the CPU, press the reset button. This will hold the CPUs reset pin low for a few microseconds. Reset is automatically performed during power-on.



CPU Card schematic.



CPU Card PCB.

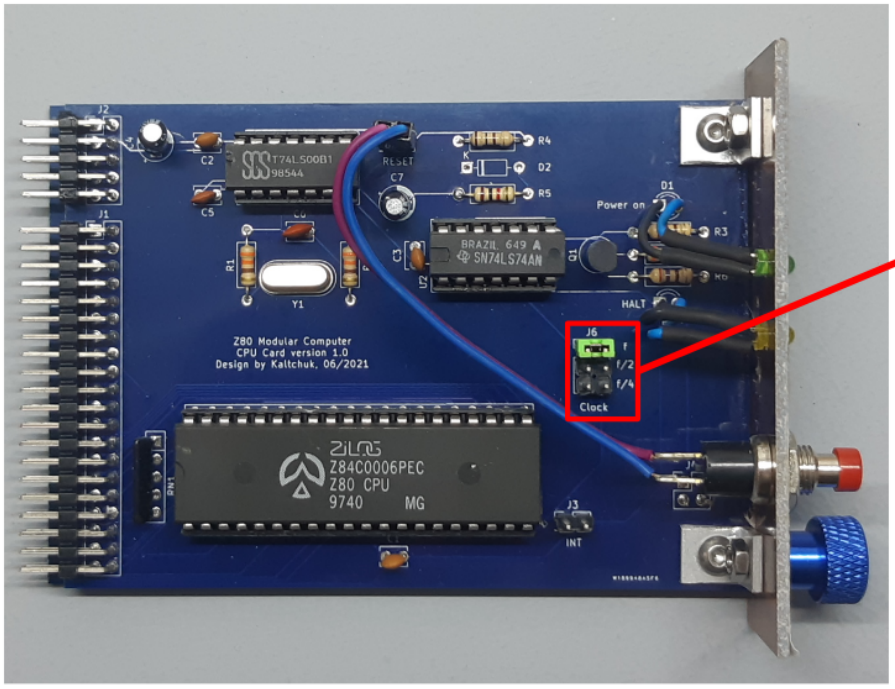
Bill of Material

Item	Qty	Reference(s)	Value	LibPart
1	3	C1, C2, C3	10n	Device:C
2	2	C4, C7	10uF	Device:CP
3	1	C5	22p	Device:C
4	1	C6	100n	Device:C
5	2	D1, D3	LED	Device:3mm LED (1 green, 1 yellow)
6	1	D2	1N4148	Diode:1N4148
7	1	J1	Signal Bus	Connector_Generic:Conn_02x20_Odd_Even _Male_90deg
8	1	J2	Power Bus	Connector_Generic:Conn_02x05_Odd_Even_Male_90deg
9	1	J3	INT	Connector_Generic:Conn_02x01_Male
10	1	J4	(void)	(void)

11	1	J5	Reset	Connector_Generic:Conn_02x01_Male
12	1	J6	clock selector	Connector_Generic:Conn_02x03_Odd_Even_male
13	1	Q1	BC328	Transistor_BJT:BC328
14	2	R1, R2	330R	Device:R
15	2	R3, R6	270R	Device:R
16	1	R4	100R	Device:R
17	1	R5	1k	Device:R
18	1	R7	3.3k	Device:R
19	1	RN1	4x10k	Device:R_Network04
20	1	U1	74LS00	74xx:74LS00 + socket
21	1	U2	74LS74	74xx:74LS74 + socket
22	1	U3	Z80CPU	CPU:Z80CPU + socket
23	1	Y1	4MHz	Device:Crystal_Small
24	1	PB1	push button	
25	1	PCB1	CPU PCB	
26	1	PCB2	CPU frontal PCB	
27	3	bolt	3Mx5mm allen	
28	1	knob	3Mx5mm	
29	4	nut	3M	
30	2	AL	10x10x1x8mm	aluminum profile

Configuration

Clock select (J6). The only configuration on the CPU Card is the clock frequency. According to the position of jumper J6, the CPU can run at 3 different speeds: f, f/2 or f/4; where f is the crystal's frequency. Since the original project uses an 8MHz crystal, the three possible speeds are: 8, 4 or 2 MHz. The picture shows the card configured to run at 8MHz.



CPU Card

J6: Clock Select

MEM Card

Description

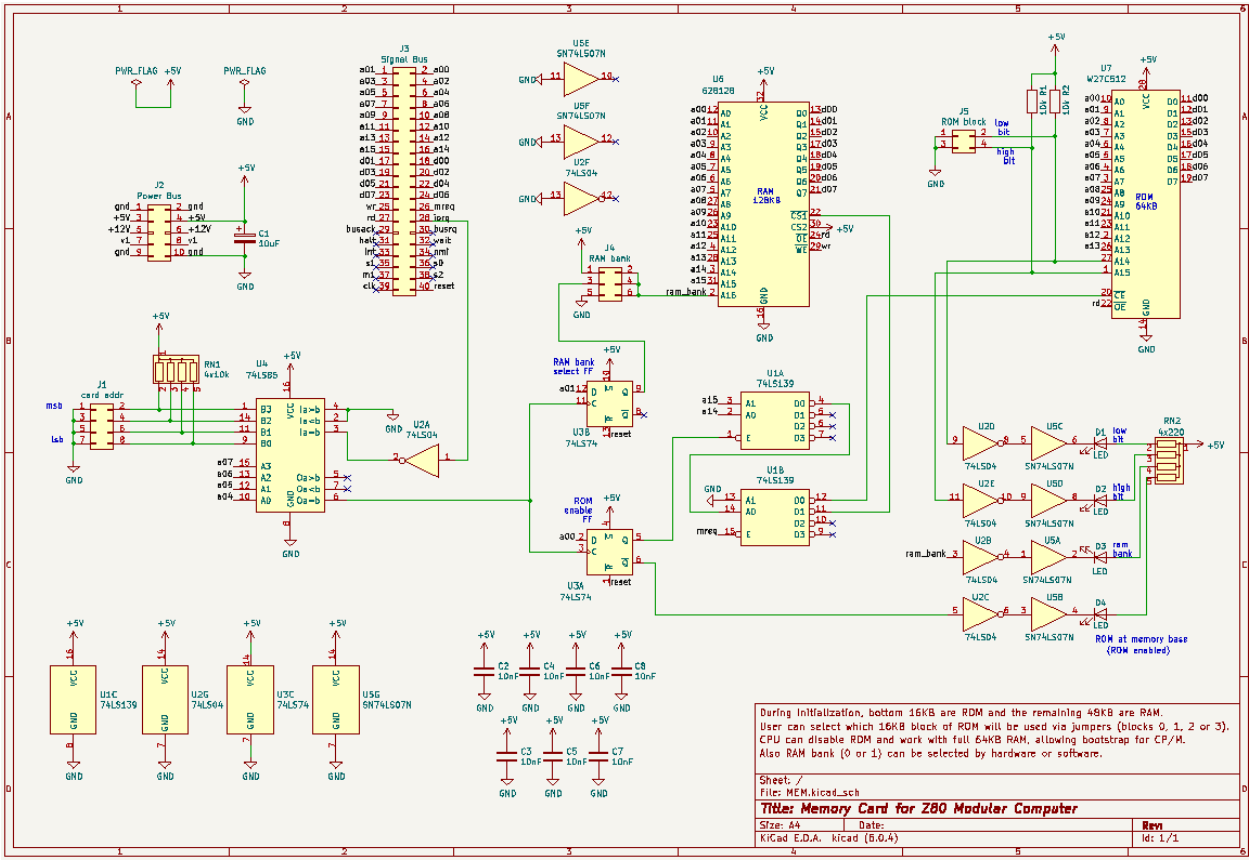
The MEM Card has 64KB of EEPROM and 128KB of SRAM. The CPU can only access 64KB of memory, so there are some circuits that handle the memory topology.

The EEPROM is divided into 4 blocks of 16KB - blocks 0, 1, 2 and 3. Block 0 contains CP/M 2.2 operating system and the BIOS. Block 1 contains Monitor 2.0 and the BIOS. Blocks 2 and 3 are empty and can be programmed by the user for other needs. The SRAM is divided into 2 banks of 64KB - banks 0 and 1.

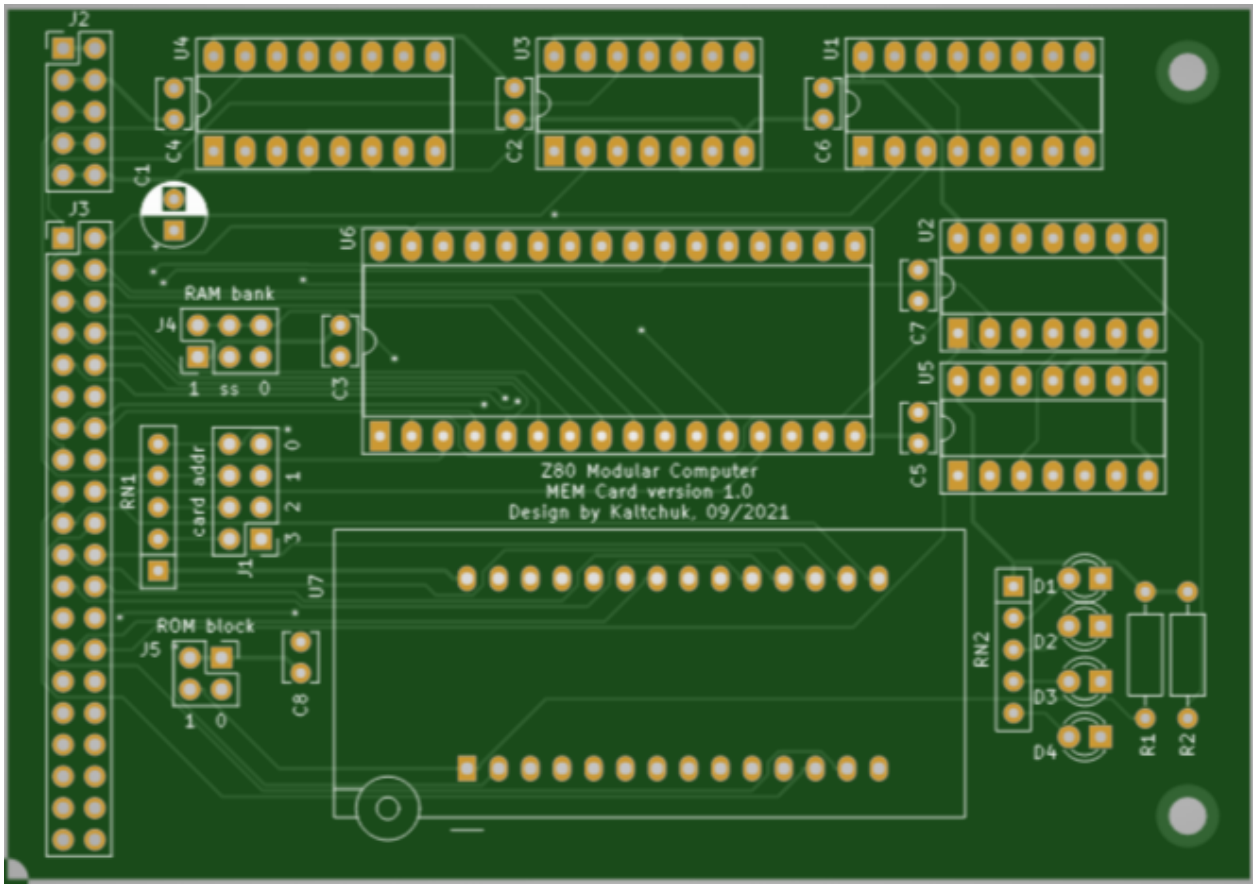
When the Proton is powered up and also after reset, the lower 16KB of memory (0000 - 3FFF) are allocated to one of the EEPROM blocks. The upper 48KB of memory (4000 - FFFF) is allocated to one of the SRAM banks. The bootloader on the EEPROM copies CP/M and the BIOS (or Monitor and the BIOS if block 1 is selected) to the top of the SRAM (D000 - FFFF) and disables the EEPROM, transforming all 64KB in SRAM.

The frontal has two LEDs to indicate which ROM block is selected and one LED to indicate if the ROM is active, i.e. if the bottom 16KB of memory is being allocated to the ROM. As soon as the BIOS starts executing it disables the ROM and the LED will turn off. There is also one LED that indicates which RAM bank is being used.

As the Proton Z80MC is intended for experimentation, the EEPROM is placed on a ZIF socket to ease insertion/removal for reprogramming.



MEM Card schematic.



MEM Card PCB.

Bill of Material

Item	Qty	Reference(s)	Value	LibPart
1	1	C1	10uF	Device:CP
2	7	C2, C3, C4, C5, C6, C7, C8	10nF	Device:C
3	4	D1, D2, D3, D4	LED	Device:3mm LED (3 red, 1 yellow)
4	1	J1	card addr	Connector_Generic:Conn_02x04_Odd_Even_Male
5	1	J2	Power Bus	Connector_Generic:Conn_02x05_Odd_Even_Male_90deg
6	1	J3	Signal Bus	Connector_Generic:Conn_02x20_Odd_Even_Male_90deg
7	1	J4	RAM bank	Connector_Generic:Conn_02x03_Odd_Even_Male
8	1	J5	ROM block	Connector_Generic:Conn_02x02_Odd_Even_Male

9	2	R1, R2	10k	Device:R
10	1	RN1	4x10k	Device:R_Network04
11	1	RN2	4x220	Device:R_Network04
12	1	U1	74LS139	74xx:74LS139 + socket
13	1	U2	74LS04	74xx:74LS04 + socket
14	1	U3	74LS74	74xx:74LS74 + socket
15	1	U4	74LS85	74xx:74LS85 + socket
16	1	U5	SN74LS07N	74xx:SN74LS07N + socket
17	1	U6	628128	Memory_RAM:628128_DIP32_SSOP32 + socket
18	1	U7	W27C512	Memory_EPROM:27C512 + ZIF socket
19	1	PCB1	MEM PCB	
20	1	PCB2	MEM frontal PCB	
21	3	bolt	3Mx5mm allen	
22	1	knob	3Mx5mm	
23	4	nut	3M	
24	2	AL	10x10x1x8mm	aluminum profile

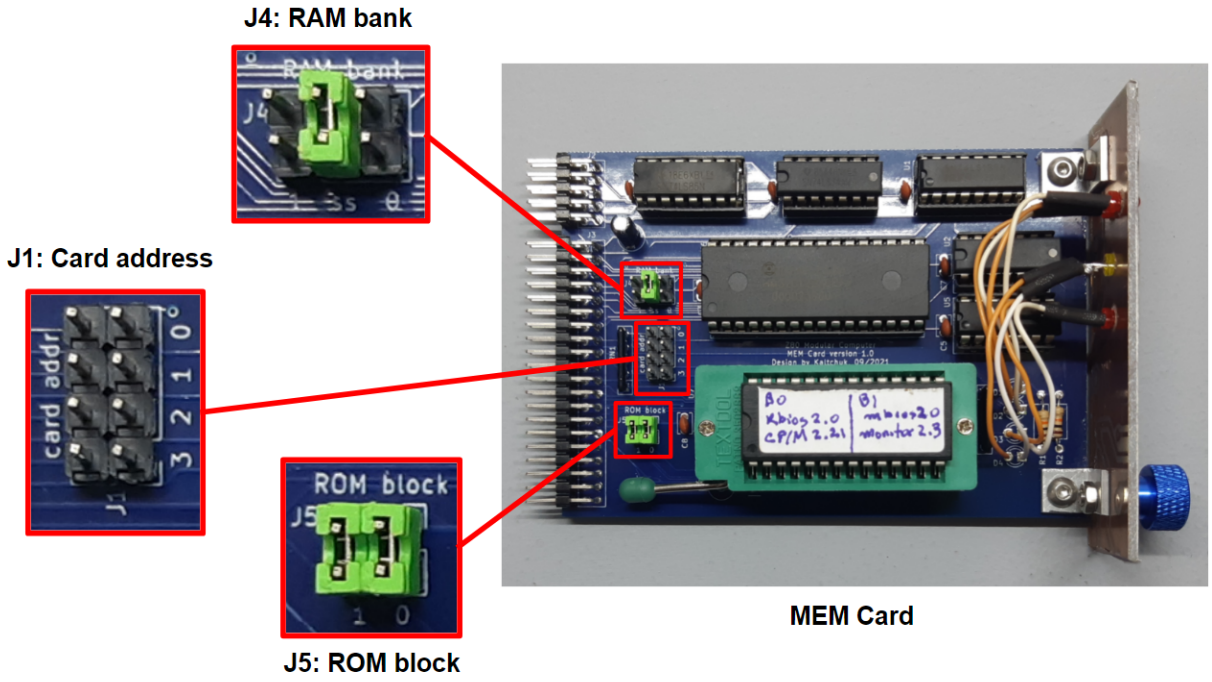
Configuration

There are three configurations needed on the MEM Card.

Card address (J1). This is the address that the CPU will use to communicate with this card via IN/OUT instructions. It is a nibble from 0 to F. No jumper means 1 and a jumper on means 0. The picture shows the card configured with address F.

RAM bank (J4). Jumper 4 selects between bank 0, bank 1 or software selectable (ss) by the CPU. The picture shows the card configured as RAM bank software selectable.

ROM block (J5). Jumper 5 selects which block will be used to boot. No jumper means 1 and a jumper on means 0. The picture shows the card configured to boot from block 0..



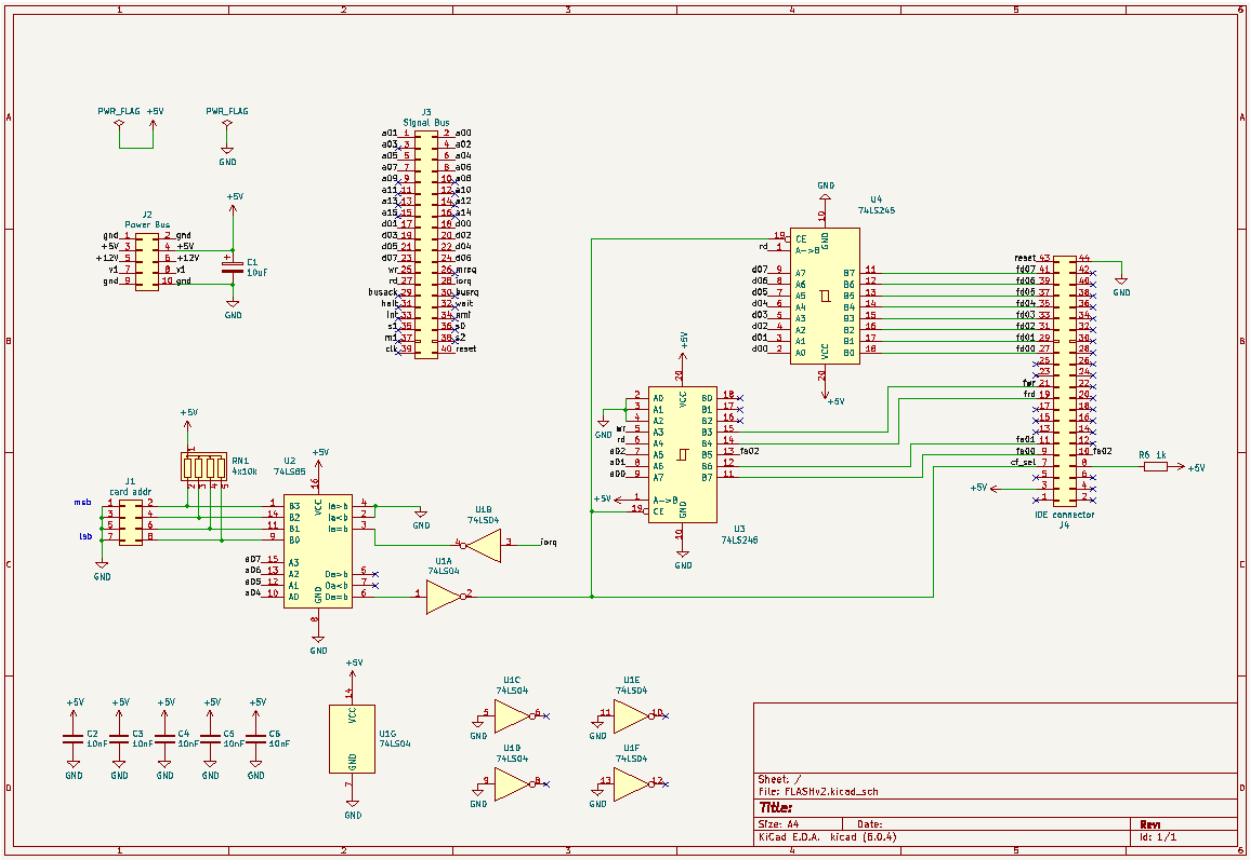
FLASH Card

Description

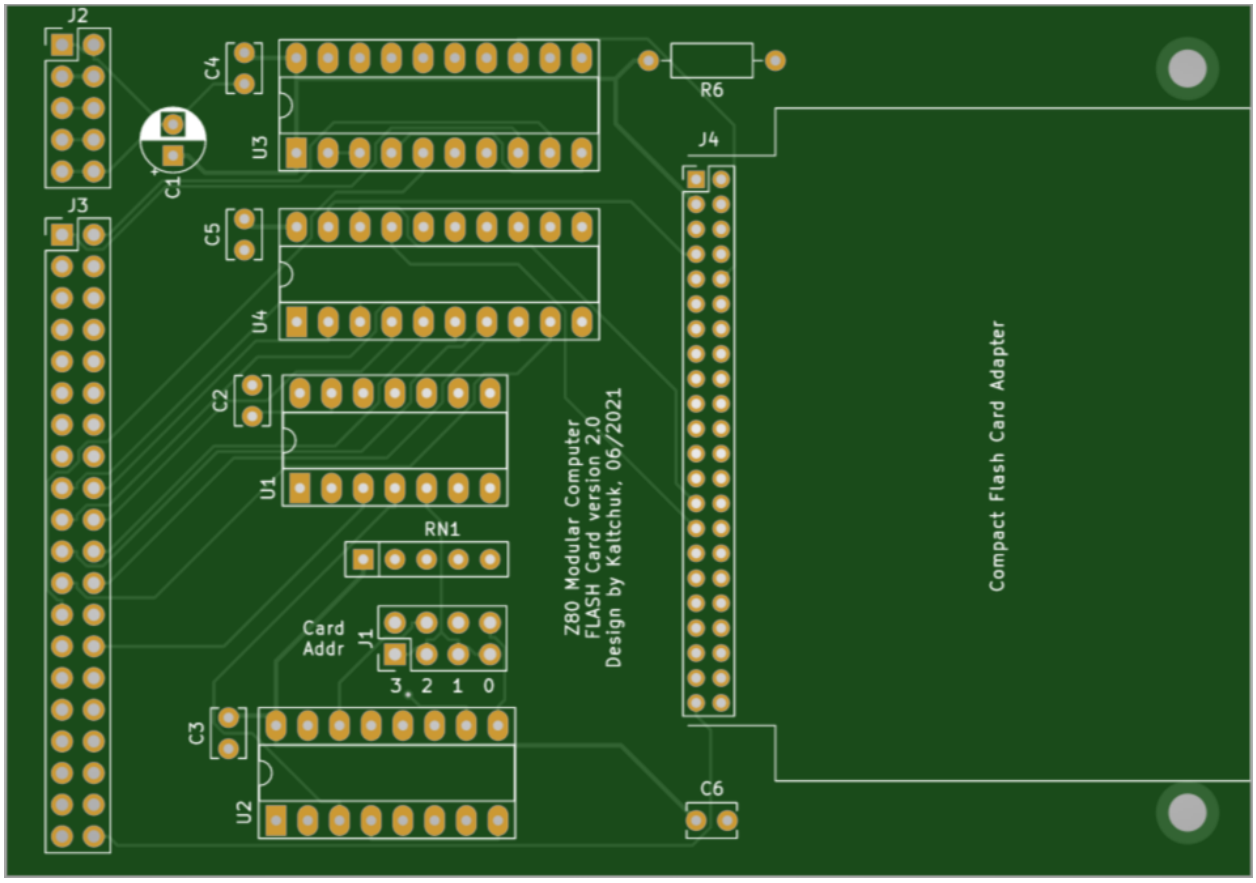
Since CP/M requires at least one floppy or hard drive to work, the FLASH Card emulates sixteen drives, each one containing 16MB. The compact flash must be formatted for use with CP/M and will contain several softwares and utilities. The backup of drive a: contains a text file named DISK.MAP that shows the summary of each drive's content.

The compact flash card seems to be very susceptible to noise on the signal lines, so version 2 of FLASH Card uses 74LS245 (octal bus transceiver with 3-state outputs) to improve the signal quality.

The frontal has an LED on the flash card to IDE adapter that indicates flash card activity (read/write).



FLASH Card schematic.



FLASH Card PCB.

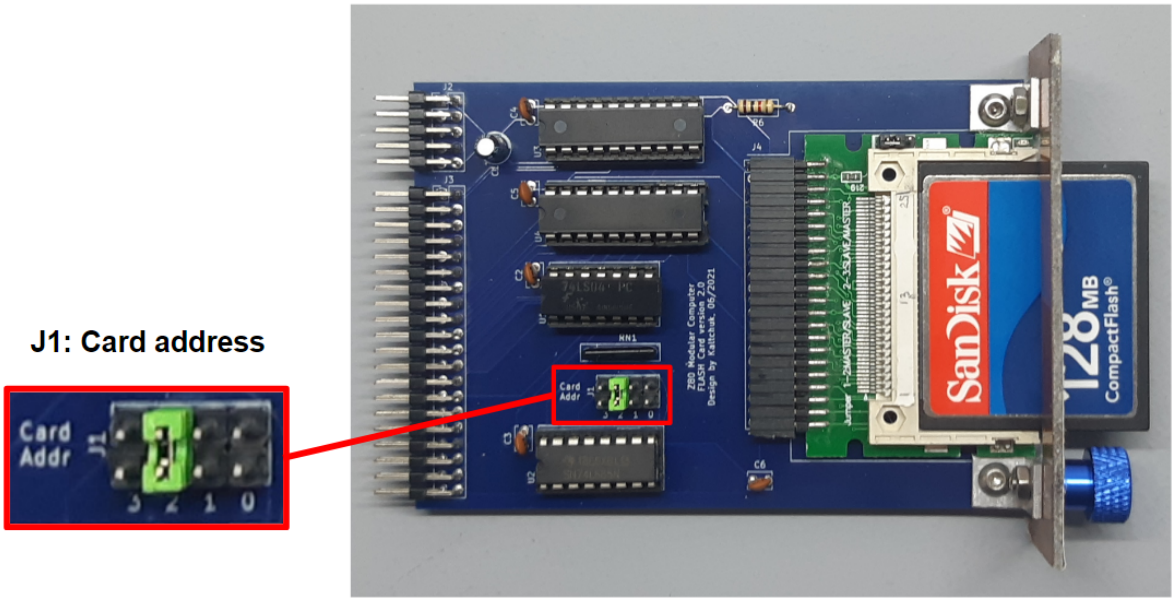
Bill of Material

Item	Qty	Reference(s)	Value	LibPart
1	1	C1	10uF	Device:CP
2	5	C2, C3, C4, C5, C6	10nF	Device:C
3	1	J1	card addr	Connector_Generic:Conn_02x04_Odd_Even_Male
4	1	J2	Power Bus	Connector_Generic:Conn_02x05_Odd_Even_Male_90deg
5	1	J3	Signal Bus	Connector_Generic:Conn_02x20_Odd_Even_Male_90deg
6	1	J4	IDE connector	Connector_Generic:Conn_02x22_Odd_Even_2.00mm_Female_90deg
7	1	R6	1k	Device:R

8	1	RN1	4x10k	Device:R_Network04
9	1	U1	74LS04	74xx:74LS04 + socket
10	1	U2	74LS85	74xx:74LS85 + socket
11	2	U3, U4	74LS245	74xx:74LS245 + socket
12		ADAPT	CF-IDE44	Compact flash to IDE44 adapter
13	1	PCB1	FLASHv2 PCB	
14	1	PCB2	FLASH frontal PCB	
15	3	bolt	3Mx5mm allen	
16	1	knob	3Mx5mm	
17	4	nut	3M	
18	2	AL	10x10x1x8mm	aluminum profile

Configuration

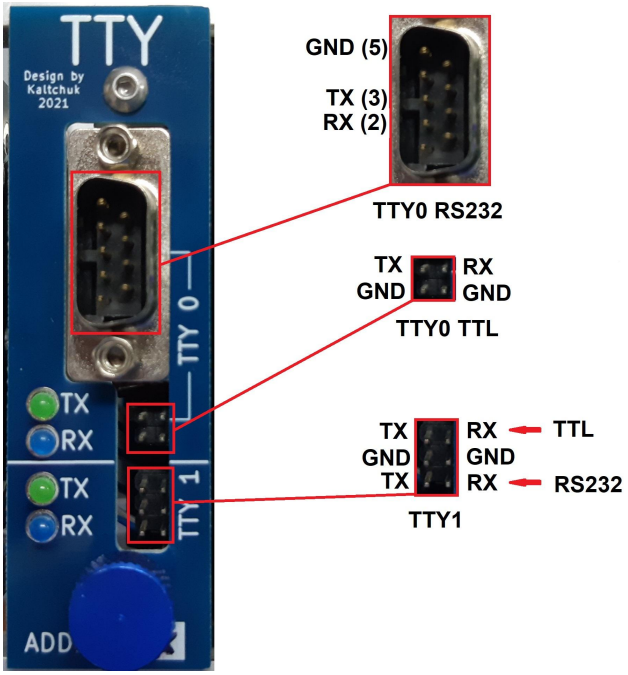
Card address (J1). The only configuration on the FLASH Card is the card's address. This is the address that the CPU will use to communicate with this card via IN/OUT instructions. It is a nibble from 0 to F. No jumper means 1 and a jumper on means 0. The picture shows the card configured with address B.



FLASH Card

TTY Card

Description



The first attempt at a serial communication card was called USART Card and was implemented via an Intel 8251 USART. It had only one serial port and a maximum baud rate of 38400bps. It used an interrupt to signal the CPU that a byte had arrived. This interrupt also generated a conflict and did not allow some softwares like SID, ZSID and DDT to run. I decided it was time to try something else. Mainly, I had two options: a) use the Zilog SIO or, b) use a non-orthodox approach. When I thought of using a microcontroller I new that it would violate one of the initial goals - “use mainly technologies available in the 1980's”; but, on the other hand, there is another goal that says: “It also has to be a hardware/software development platform for students, hobbyists and enthusiasts”. Based on this, I decided to develop a card with two serial ports, each one controlled by an independent microcontroller; and based on my previous experience with Arduino, I chose the ATmega328.

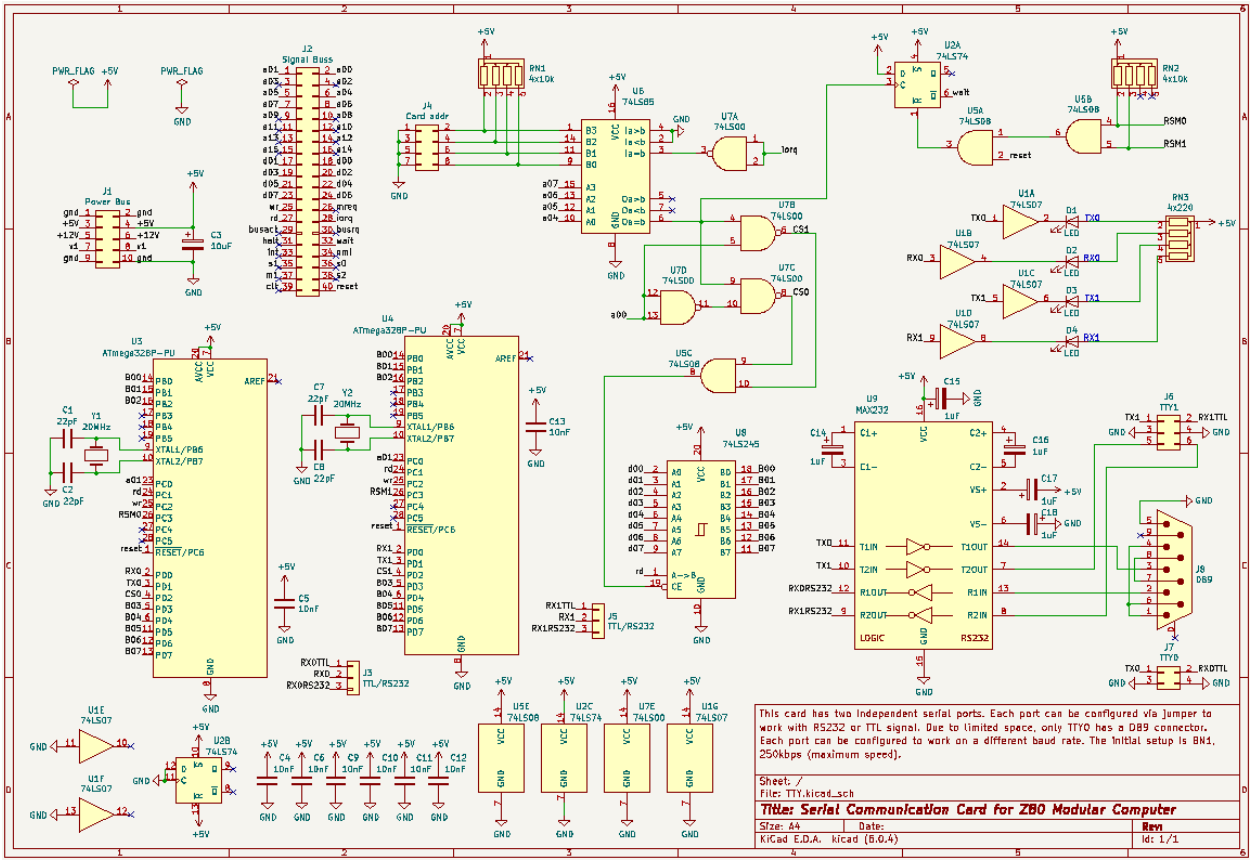
The TTY Card provides the primary means for the Z80MC to communicate with the user and other serial peripherals. It has two independent serial ports - TTY0 and TTY1. Each port can be configured to work at 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, 125000 or 250000 bps via software on CP/M. Note that there is no 115200 bps option due to the fact that the baud rate generated with the 24MHz of the microcontroller’s clock has an unacceptable baud rate error. To change the TTY baud rate, run **`TTYBAUD.COM`**; select the port and the desired baud rate. Keep in mind that after reset, both ports will start at 250kbps, 8 bits, no parity and 1 stop bit. Besides the baud rate, all parameters can be changed but this requires modifying the firmware on the ATmega328 microcontrollers.

Both ports can be configured by hardware to use RS232 or TTL level (see Configuration below). Due to limited space, only TTY0 has a DB9 connector.

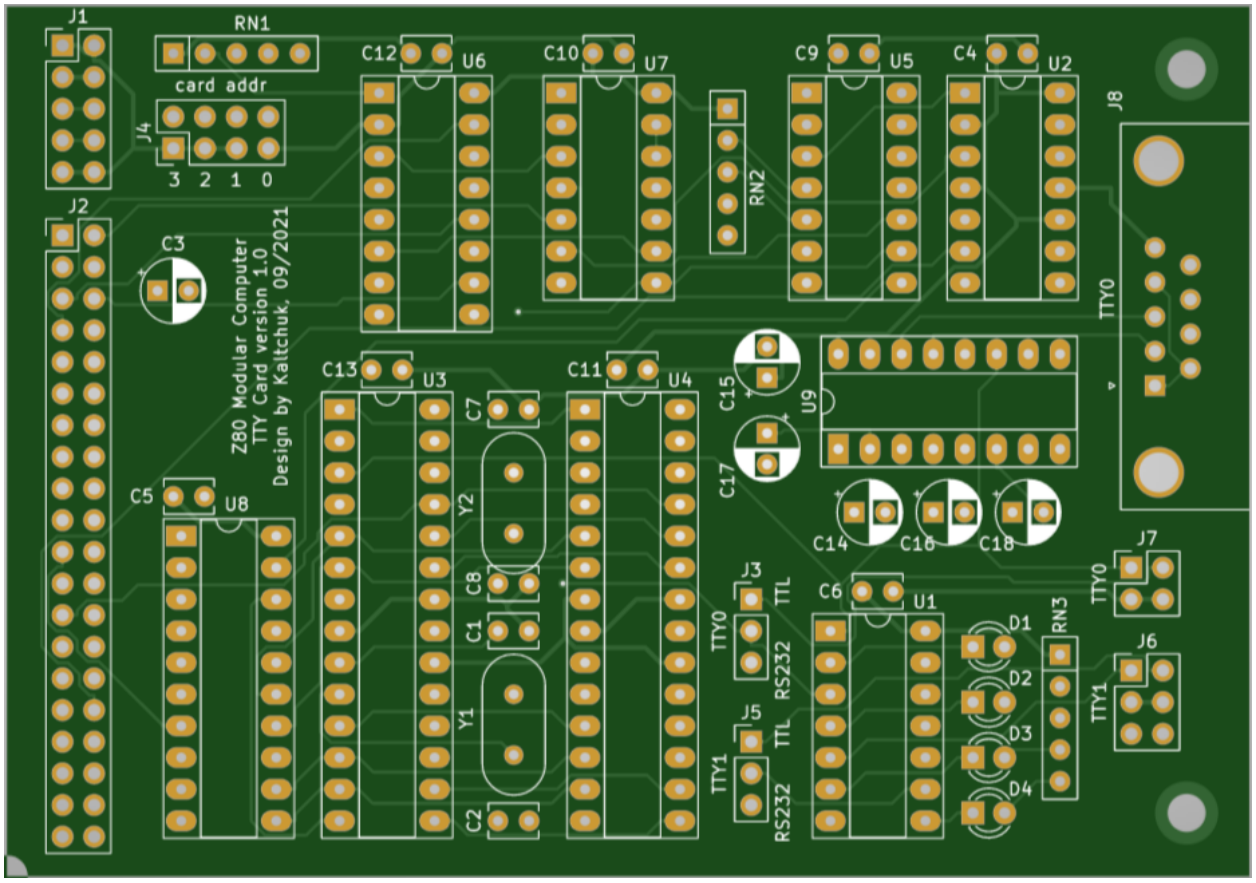
Each port has a 128 byte circular input buffer, which means that data will be lost if the CPU does not fetch the data before the buffer is full. The card does not inform the CPU when new data has arrived; so it is the program’s responsibility to check for new data in the buffers.

The frontal has 4 LEDs, two for each port, the green LEDs are for RX and the blue LEDs are for TX.

The first address of the card is used by TTY0 and the second by TTY1. For example, let's say that the card is configured with address “C”. In this case, the CPU will communicate with TTY0 using input/output on address C0 and with TTY1 on address C1.



TTY Card schematic.



TTY Card PCB.

Bill of Material

Item	Qty	Reference(s)	Value	LibPart
1	4	C1, C2, C7, C8	22pF	Device:C
2	1	C3	10uF	Device:C_Polarized
3	8	C4, C5, C6, C9, C10, C11, C12, C13	10nF	Device:C
4	5	C14, C15, C16, C17, C18	1uF	Device:C_Polarized
5	4	D1, D2, D3, D4	LED	Device:3mm LED (2 green, 2 blue)
6	1	J1	Power Bus	Connector_Generic:Conn_02x05_Odd_Even_Male_90deg
7	1	J2	Signal Buss	Connector_Generic:Conn_02x20_Odd_Even_Male_90d

				eg
8	2	J3, J5	TTL/RS232	Connector_Generic:Conn_01x03_Male
9	1	J4	Card addr	Connector_Generic:Conn_02x04_Odd_Even_Male
10	1	J6	TTY1	Connector_Generic:Conn_02x03_Odd_Even_Male
11	1	J7	TTY0	Connector_Generic:Conn_02x02_Odd_Even_Male
12	1	J8	DB9	Connector:DB9_Male_MountingHoles
13	2	RN1, RN2	4x10k	Device:R_Network04
14	1	RN3	4x220	Device:R_Network04
15	1	U1	74LS07	74xx:74LS07 + socket
16	1	U2	74LS74	74xx:74LS74 + socket
17	2	U3, U4	ATmega328P-PU	TTY-rescue:ATmega328P-PU-MCU_Microchip_ATmega + socket
18	1	U5	74LS08	74xx:74LS08 + socket
19	1	U6	74LS85	74xx:74LS85 + socket
20	1	U7	74LS00	74xx:74LS00 + socket
21	1	U8	74LS245	74xx:74LS245 + socket
22	1	U9	MAX232	Interface_UART:MAX232 + socket
23	2	Y1, Y2	20MHz	Device:Crystal
26	1	PCB1	TTY PCB	
27	1	PCB2	TTY frontal PCB	
28	3	bolt	3Mx5mm allen	
29	1	knob	3Mx5mm	
30	4	nut	3M	
31	2	AL	10x10x1x8mm	aluminum profile

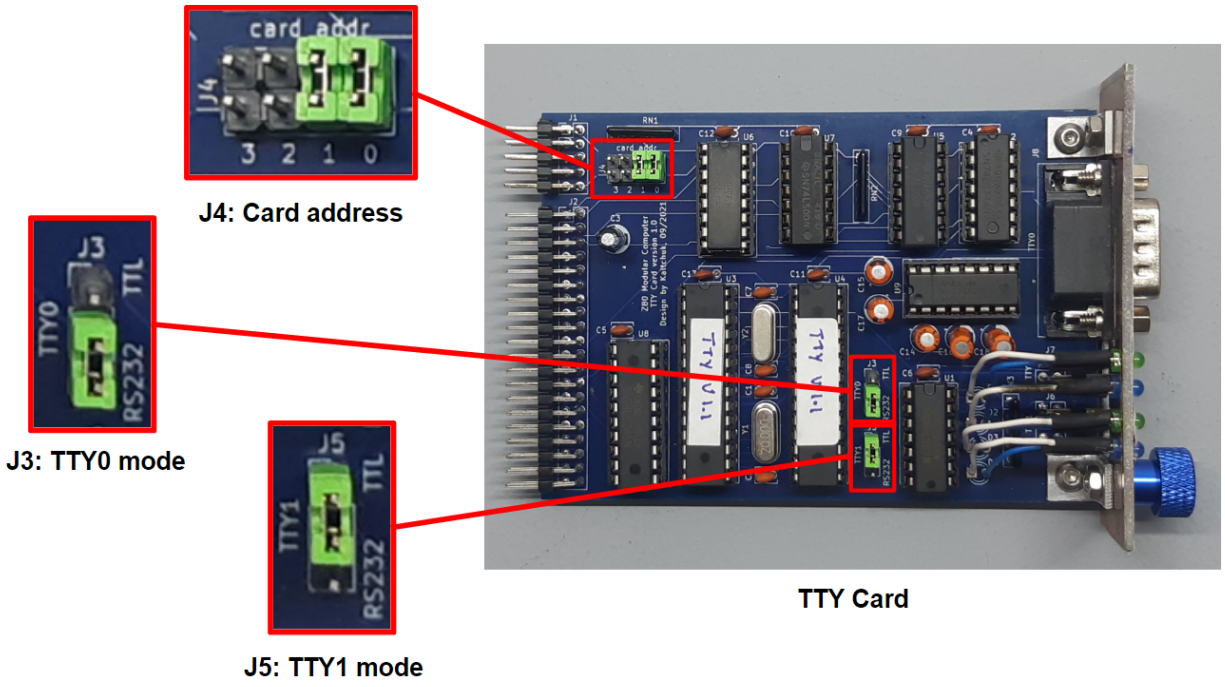
Configuration

There are three configurations needed on the TTY Card.

Card address (J4). This is the address that the CPU will use to communicate with this card via IN/OUT instructions. It is a nibble from 0 to F. No jumper means 1 and a jumper on means 0. The picture shows the card configured with address C.

TTY0 mode (J3). This jumper selects if port TTY0 will communicate using RS232 or TTL level. The picture shows TTY0 configured as RS232.

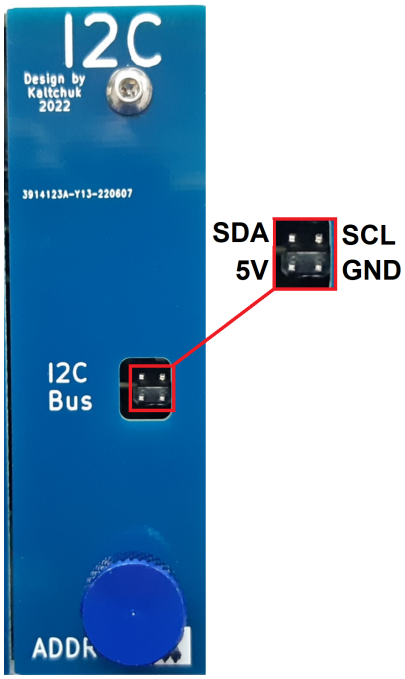
TTY1 mode (J5). This jumper selects if port TTY1 will communicate using RS232 or TTL level. The picture shows TTY1 configured asTTL level.



I2C Card

Description

The I2C Card is an easy way to interface the CPU Card with I2C peripherals. The card has an Atmel AT89C4051 microcontroller which receives high level read/write commands from the CPU and transforms them to I2C communication protocol. Besides the microcontroller, the card has an 8KB EEPROM and a real time clock with a backup battery (CR2032).



The card also has a 2 x 4 cm prototyping area for more I2C components. The frontal panel has a four pin connector for external I2C devices.

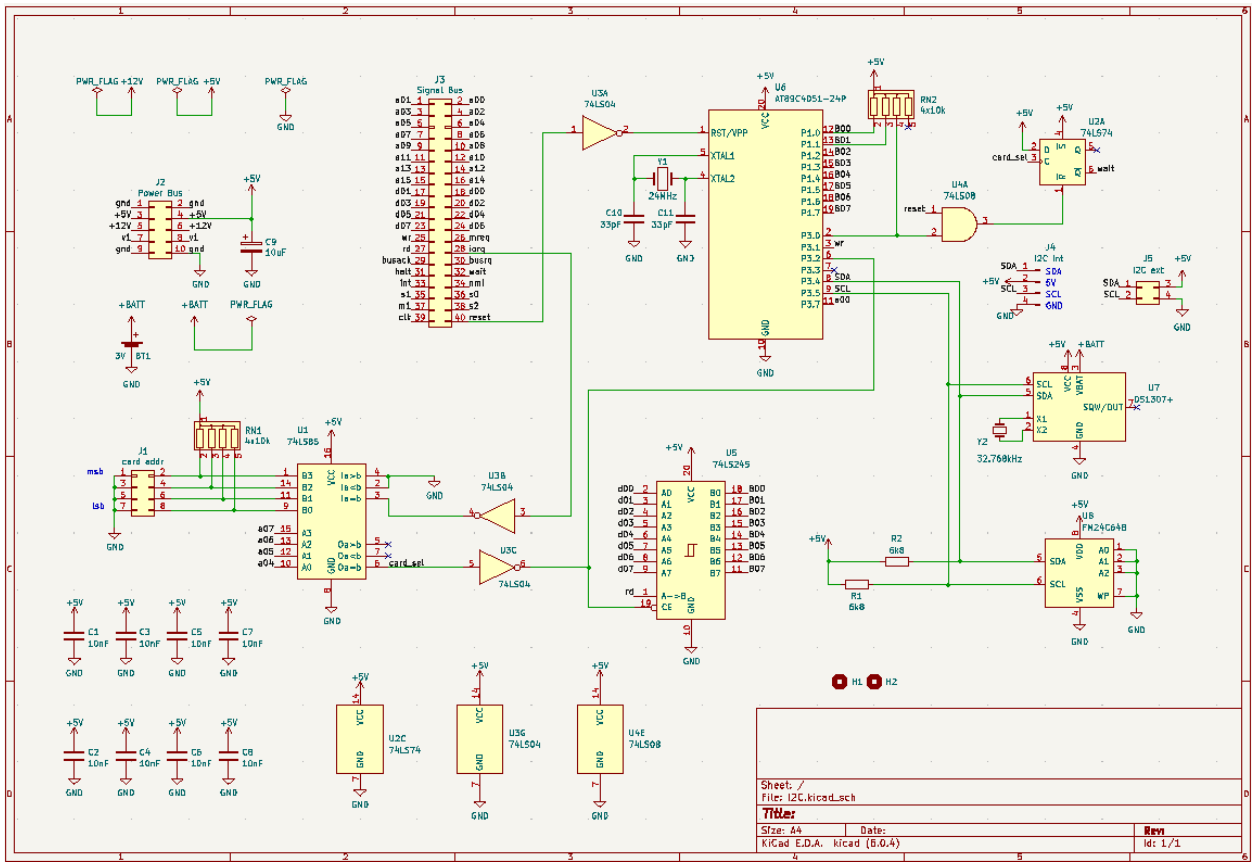
Communication with the I2C Card can be easily implemented in C, using the functions in the header file **I2C2.H**, in drive c:. Of course, a minimum knowledge of I2C protocol is necessary for this task. There are three functions in this header file - write, read and read random; with the following syntax:

```
I2Cwr(i2c_card, slave, size_addr, addr, num_bytes, buf)
char    i2c_card, slave, size_addr, num_bytes, *buf;
int     addr;

I2Crd(i2c_card, slave, num_bytes, buf)
char    i2c_card, slave, num_bytes, buf[];

I2Crr(i2c_card, slave, size_addr, addr, num_bytes, buf)
char    i2c_card, slave, size_addr, num_bytes, buf[];
int     addr;
```

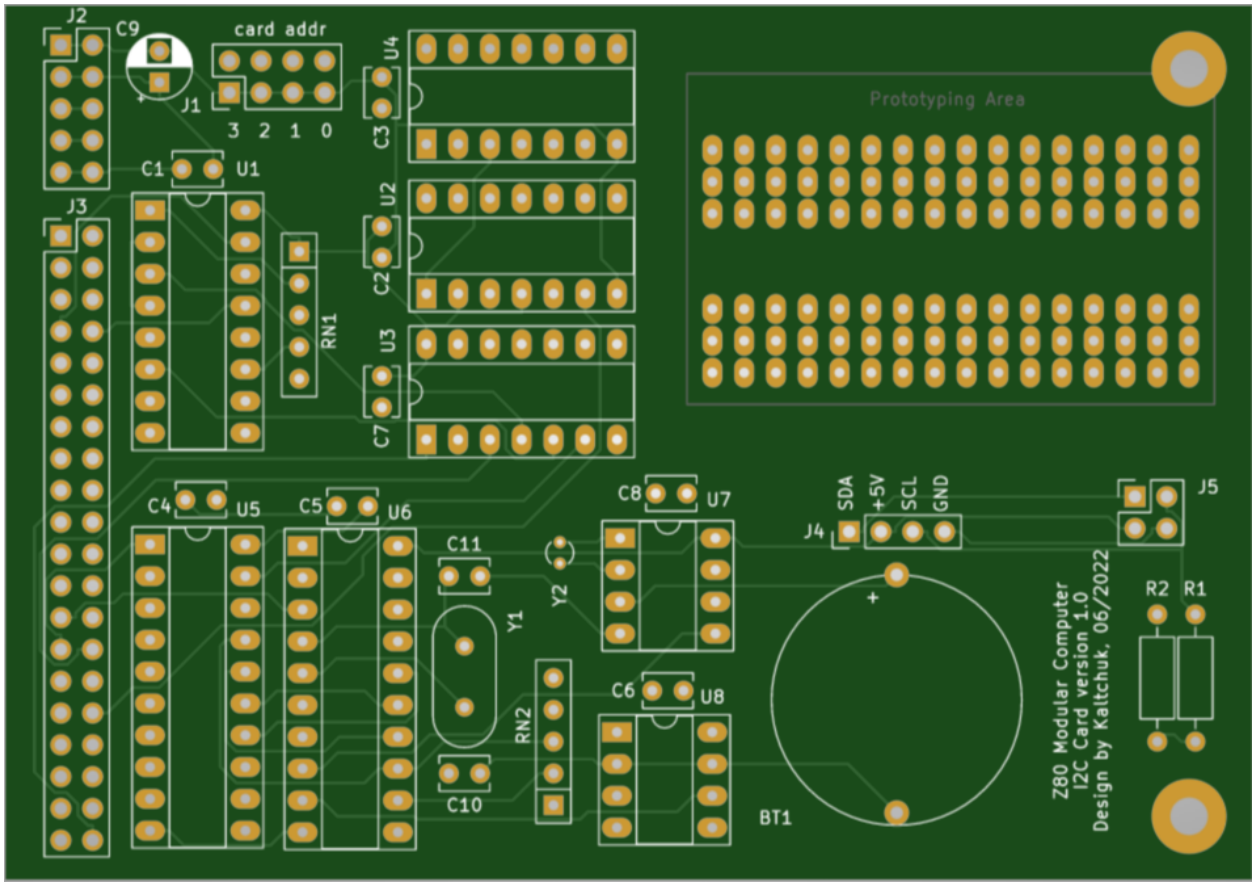
As an example of the I2C Card, there is a program called **CLOCK.COM**, in drive c: that can read and set time and date.

```
C>clock
Set/get time and date from RTC

Commands: t?           Get time.
          t=hh:mm:ss   Set time.
          d?           Get date.
```

```
          d=w dd/mm/yy Set date.
          q            Quit program.

# t?
  20:29:53
# d?
  Mon 11/07/22
# q

>C
```



I2C Card schematic.



I2C Card PCB.

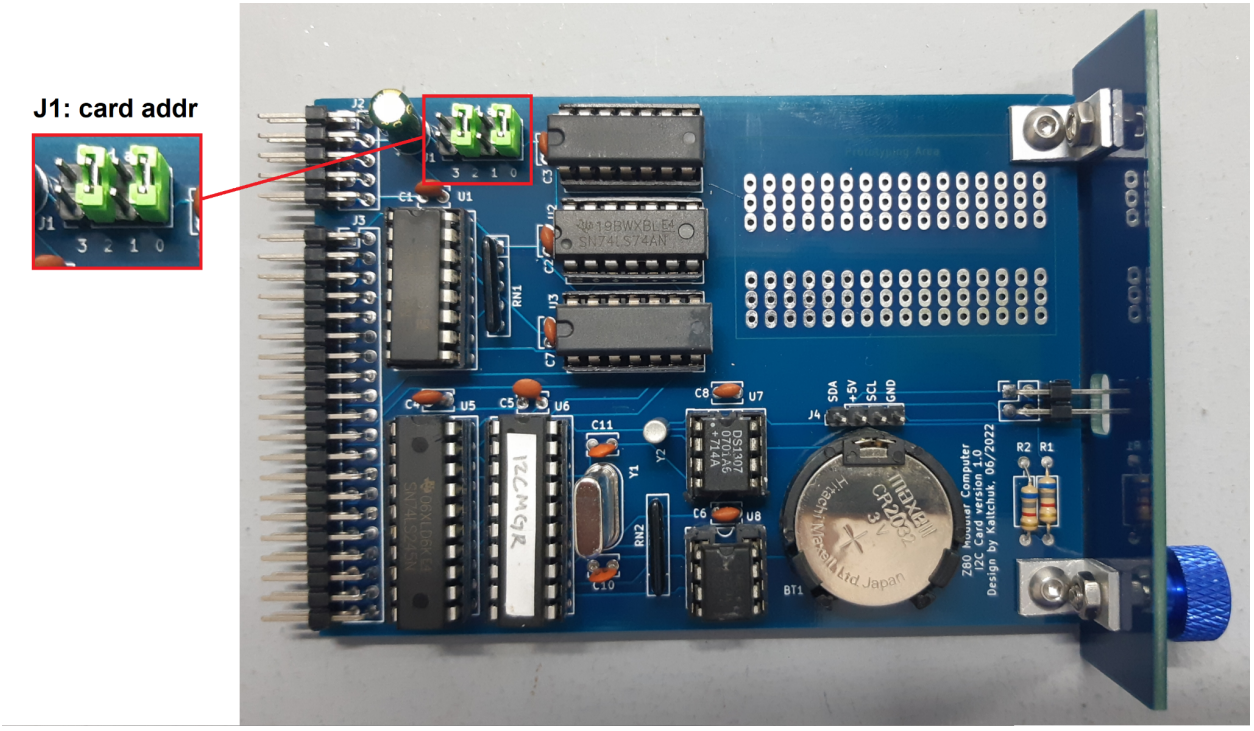
Bill of Material

Item	Qty	Reference(s)	Value	LibPart
1	1	BT1	CR2032	Device:Battery_Cell + socket
2	8	C1, C2, C3, C4, C5, C6, C7, C8	10nF	Device:C
3	1	C9	10uF	Device:CP
4	2	C10, C11	33pF	Device:C
5	2	H1, H2	MountingHole	Mechanical:MountingHole
6	1	J1	card addr	Connector_Generic:Conn_02x04_Odd_Even_Male
7	1	J2	Power Bus	Connector_Generic:Conn_02x05_Odd_Even_Male_90deg
8	1	J3	Signal Bus	Connector_Generic:Conn_02x20_Odd_Even_Male_90deg

9	1	J4	I2C int	Connector:Conn_01x04_Male
10	1	J5	I2C ext	Connector_Generic:Conn_02x02_Odd_Even_Male_90deg
11	2	R1, R2	6k8	Device:R
12	2	RN1, RN2	4x10k	Device:R_Network04
13	1	U1	74LS85	74xx:74LS85 + socket
14	1	U2	74LS74	74xx:74LS74 + socket
15	1	U3	74LS04	74xx:74LS04 + socket
16	1	U4	74LS08	74xx:74LS08 + socket
17	1	U5	74LS245	74xx:74LS245 + socket
18	1	U6	AT89C4051-24P	MCU_Microchip_8051:AT89C4051-24P + socket
19	1	U7	DS1307	Timer_RTC:DS1307 + socket
20	1	U8	FM24C64B	Memory_NVRAM:FM24C64B + socket
21	1	Y1	24MHz	Device:Crystal
22	1	Y2	32.768kHz	Device:Crystal_Small
23	1	PCB1	I2C PCB	
24	1	PCB2	I2C frontal PCB	
25	3	bolt	3Mx5mm allen	
26	1	knob	3Mx5mm	
27	4	nut	3M	
28	2	AL	10x10x1x8mm	aluminum profile

Configuration

Card address (J1). This is the address that the CPU will use to communicate with this card via IN/OUT instructions. It is a nibble from 0 to F. No jumper means 1 and a jumper on means 0. The picture shows the card configured with address A.



PSU Card

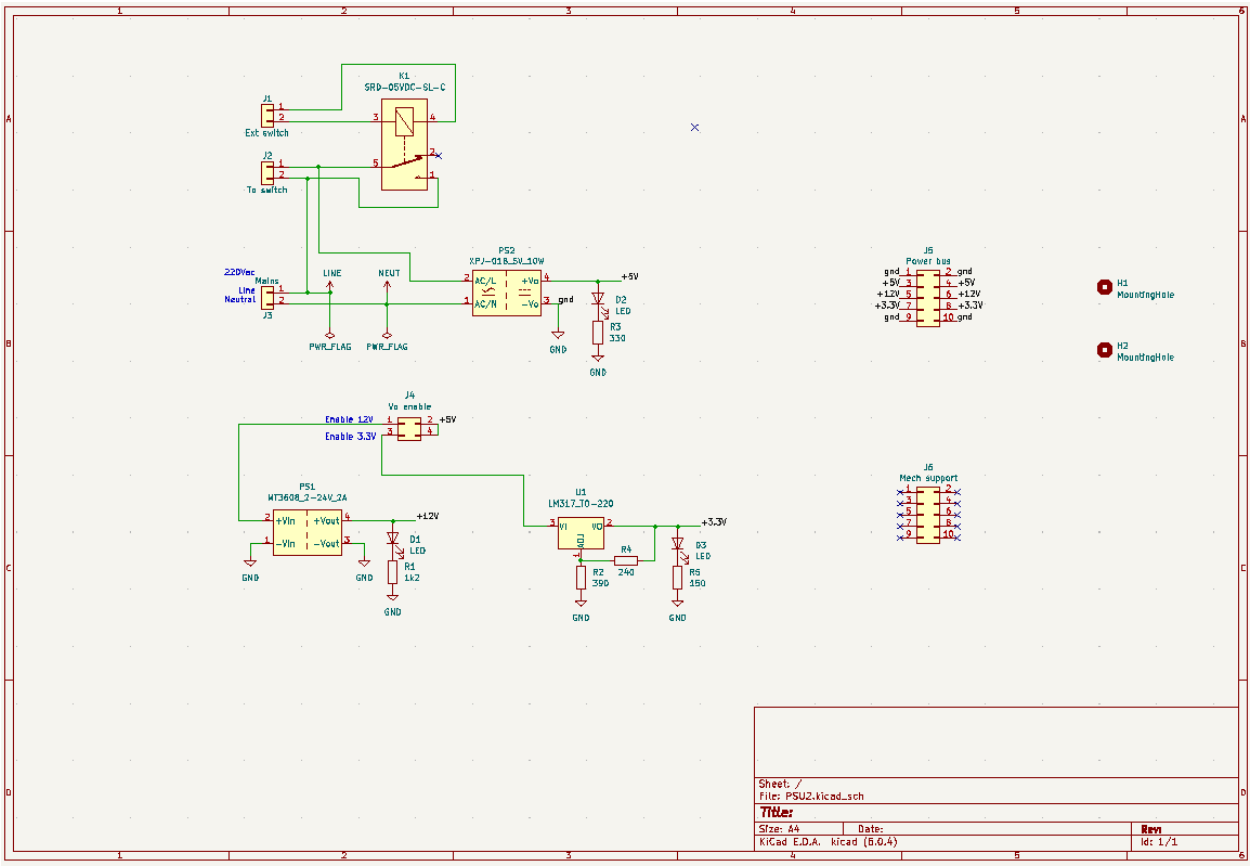
Description

The PSU Card provides the 5Vdc necessary for all other cards. It has a maximum output of 2A and has to be connected to an external 100 - 240Vac, 50/60Hz supply. The card has an input for external power-on via a 5V relay. For example, a modem can be used to turn on the PSU using a 5V signal.

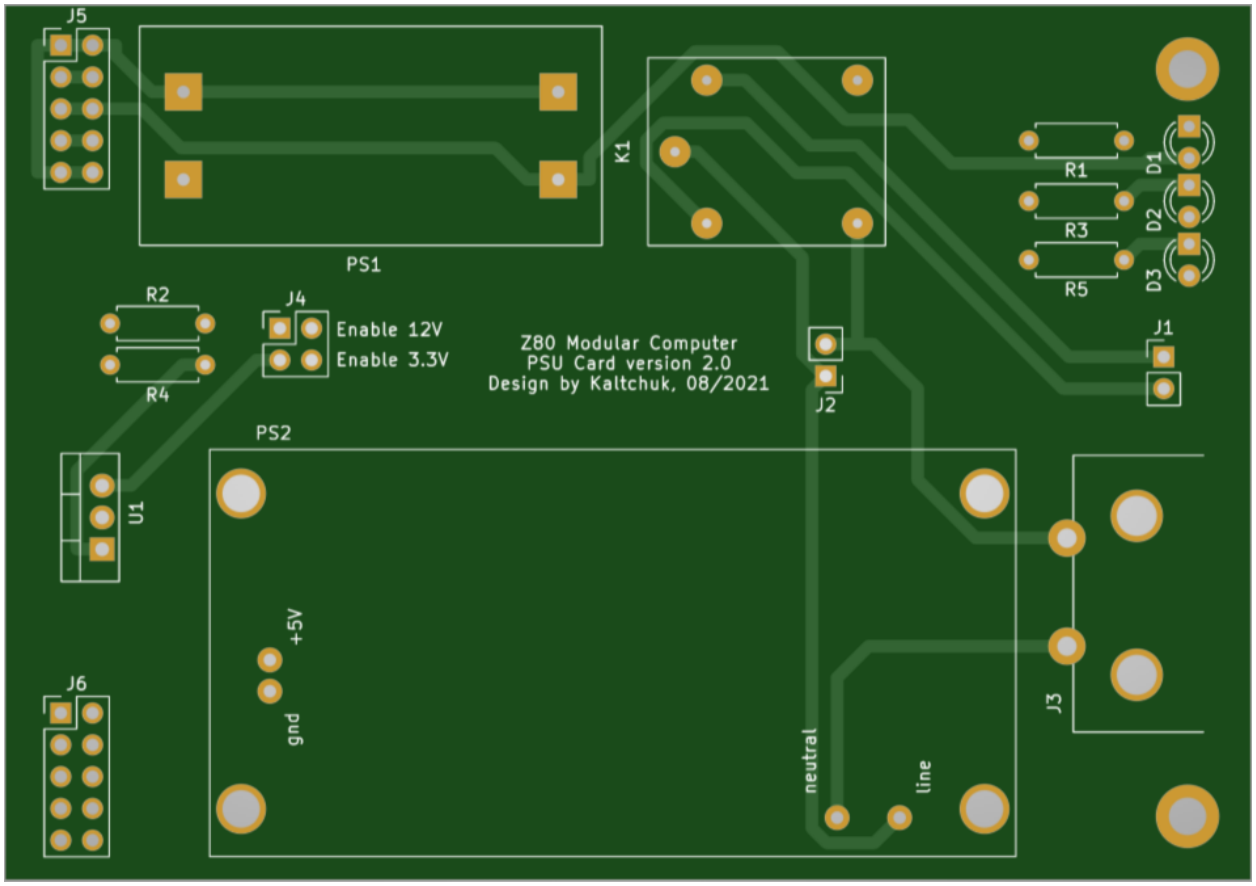
Besides 5V, the PSU Card can also provide 12V and 3.3V for other cards.

The frontal has an ON/OFF switch, a connector for the power cord and a connector for external power-on command.

There are also three LEDs, one for each voltage output - 12, 5 and 3.3V. If the LED is on, it means that the respective voltage is enabled.



PSU Card schematic.



PSU Card PCB.

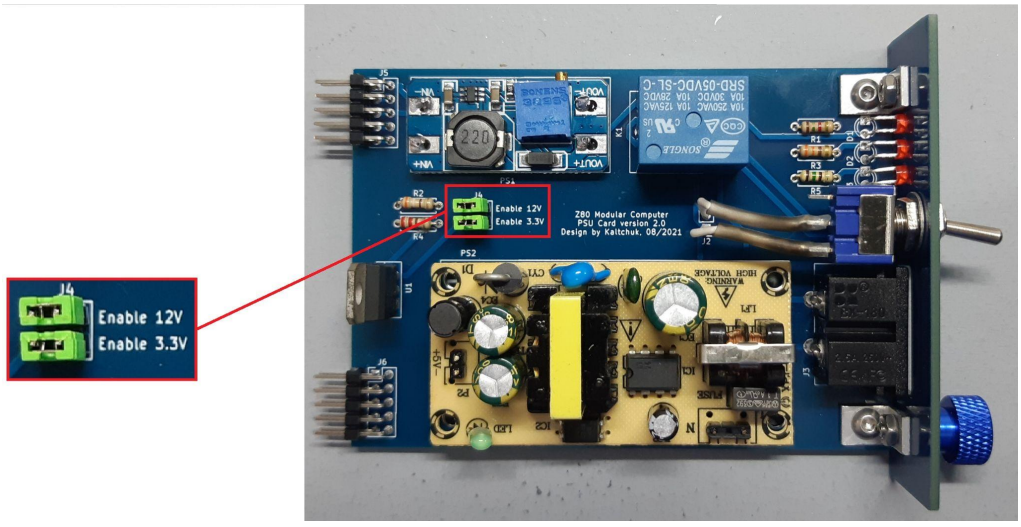
Bill of Material

Item	Qty	Reference(s)	Value	LibPart
1	3	D1, D2, D3	LED	Device:LED
2	2	H1, H2	MountingHole	Mechanical:MountingHole
3	1	J1	Ext switch	Connector_Generic:Conn_01x02_Male_90deg
4	1	J2	To switch	Connector_Generic:Conn_01x02
5	1	J3	Mains	Connector_Generic:Conn_01x02
6	1	J4	Vo enable	Connector_Generic:Conn_02x02_Odd_Even_Male
7	1	J5	Power bus	Connector_Generic:Conn_02x05_Odd_Even_Male_90deg
8	1	J6	Mech support	Connector_Generic:Conn_02x05_Odd_Even_Male_90deg
9	1	K1	SRD-05VDC-SL-C	Relay:Y14x-1C-xxDS
10	1	PS1	MT3608_2-24V_2A	Converter_DCDC:MEE1S0512SC

11	1	PS2	XPJ-01B_5V_10W	Converter_ACDC:IRM-10-5
12	1	R1	1k2	Device:R
13	1	R2	390	Device:R
14	1	R3	330	Device:R
15	1	R4	240	Device:R
16	1	R5	150	Device:R
17	1	U1	LM317_TO-220	Regulator_Linear:LM317_TO-220
18	1	PCB1	PSU PCB	
19	1	PCB2	PSU frontal PCB	
20	3	bolt	3Mx5mm allen	
21	1	knob	3Mx5mm	
22	4	nut	3M	
23	2	AL	10x10x1x8mm	aluminum profile

Configuration

Output voltage enable (J4). These jumpers allow the user to enable or disable the 3.3V and 12V output. The picture shows both voltages enabled.



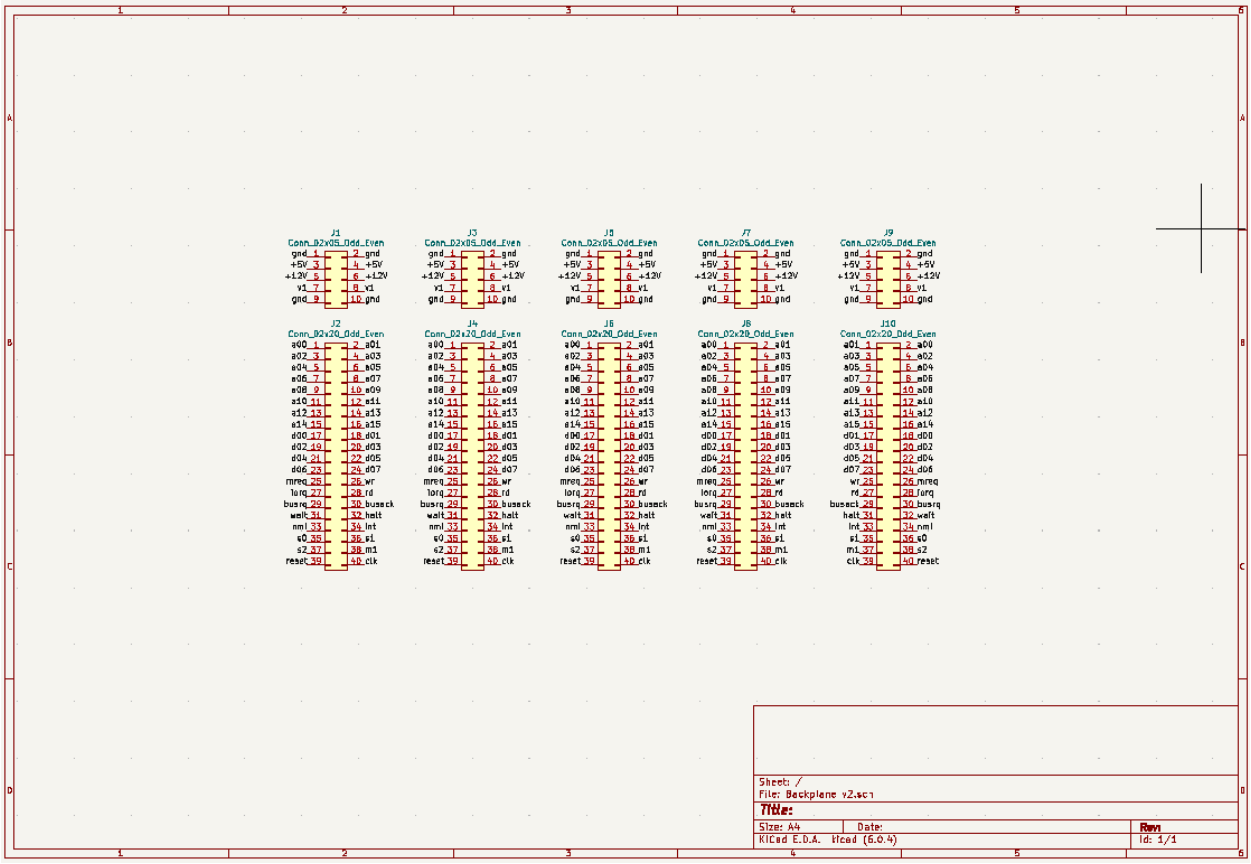
Backplane

Description

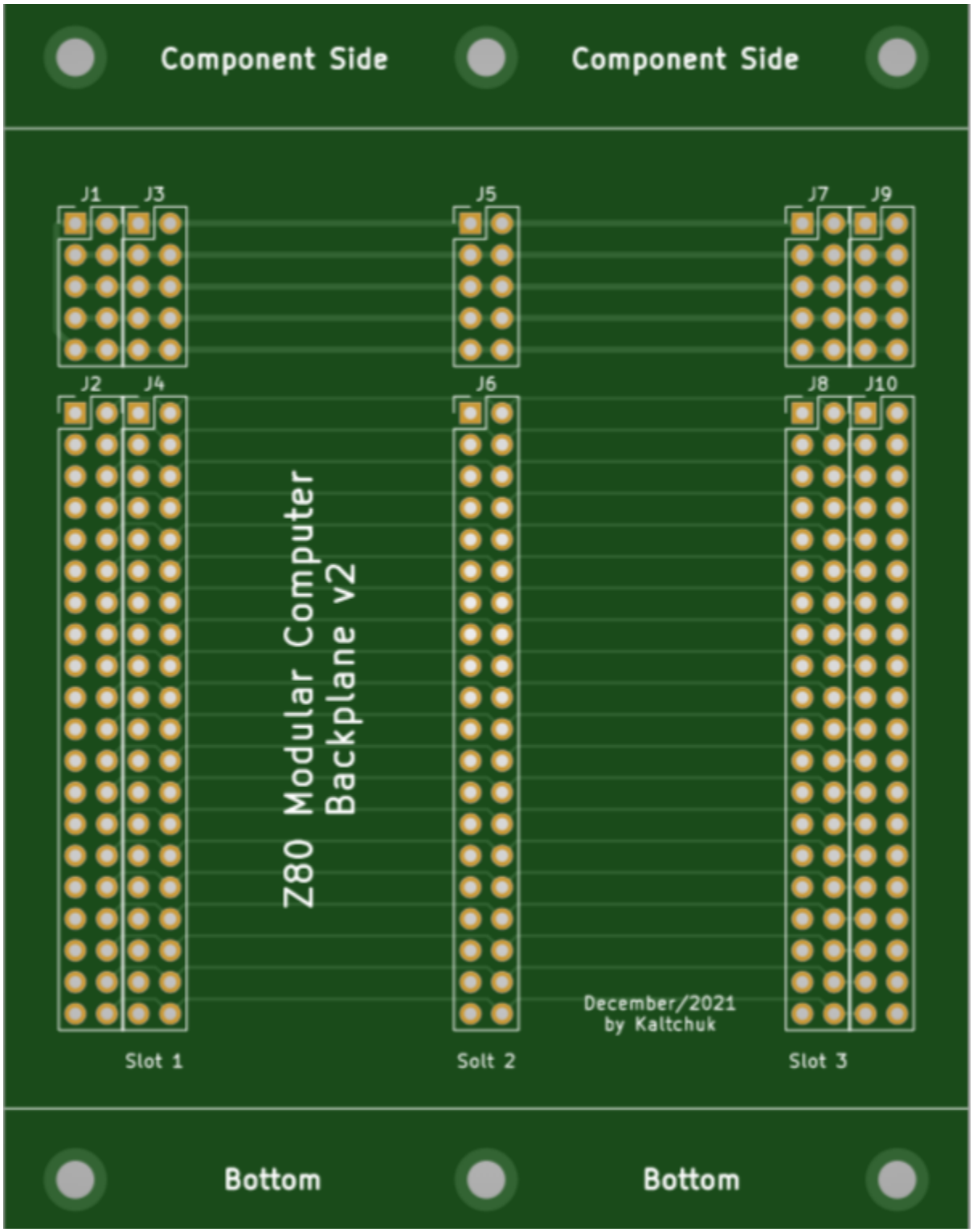
The Backplane Card is completely passive and has no electronic component, only connectors. Each Backplane card has slots for 3 modules and cascading connectors so several backplane cards can be interconnected. In order to run CP/M, a minimum of five modules are required - CPU, MEM, TTY, FLASH and PSU, so at least two backplane cards must be used. This gives a total space for 6 modules (five slots will be used and one will be empty). The recommended configuration is with three backplane cards, which gives a total of 9 slots (5 in use and 4 empty).

The backplane is divided into two buses - power bus and signal bus. The power bus contains all power lines (5V, 12V and 3.3V). Since the original project didn't foresee the need for 3.3V, this voltage is assigned to line V1.

The signal bus contains all pins of the Z80 CPU, except Vcc and GNG, plus two spare lines (S0 and S1) for user's needs. S2 is actually used by the RFSH signal of the CPU.



Backplane Schematic.



Backplane PCB.

The picture below shows a fully assembled backplane card that can be used as a central piece in a 9 slot rack. The leftmost card in the rack doesn't need connectors J1 and J2, as also the rightmost card doesn't need connectors J9 and J10.

The next picture shows a fully assembled 9 slot rack composed of 3 backplane cards.

Bill of Material

Item	Qty	Reference(s)	Value	LibPart
1	1	J1	Conn_02x05_Odd_Even	Connector_Generic:Conn_02x05_Odd_Even_Male_90deg
2	1	J2	Conn_02x20_Odd_Even	Connector_Generic:Conn_02x20_Odd_Even_Male_90deg
3	3	J3, J5, J7	Conn_02x05_Odd_Even	Connector_Generic:Conn_02x05_Odd_Even_Female
4	3	J4, J6, J8	Conn_02x20_Odd_Even	Connector_Generic:Conn_02x20_Odd_Even_Female
5	1	J9	Conn_02x05_Odd_Even	Connector_Generic:Conn_02x05_Odd_Even_Female_90deg
6	1	J10	Conn_02x20_Odd_Even	Connector_Generic:Conn_02x20_Odd_Even_Female_90deg
7	1	PCB1	Backplane PCB	
8	4	Bolt*	3Mx5mm allen	
9	4	Nut*	3M	

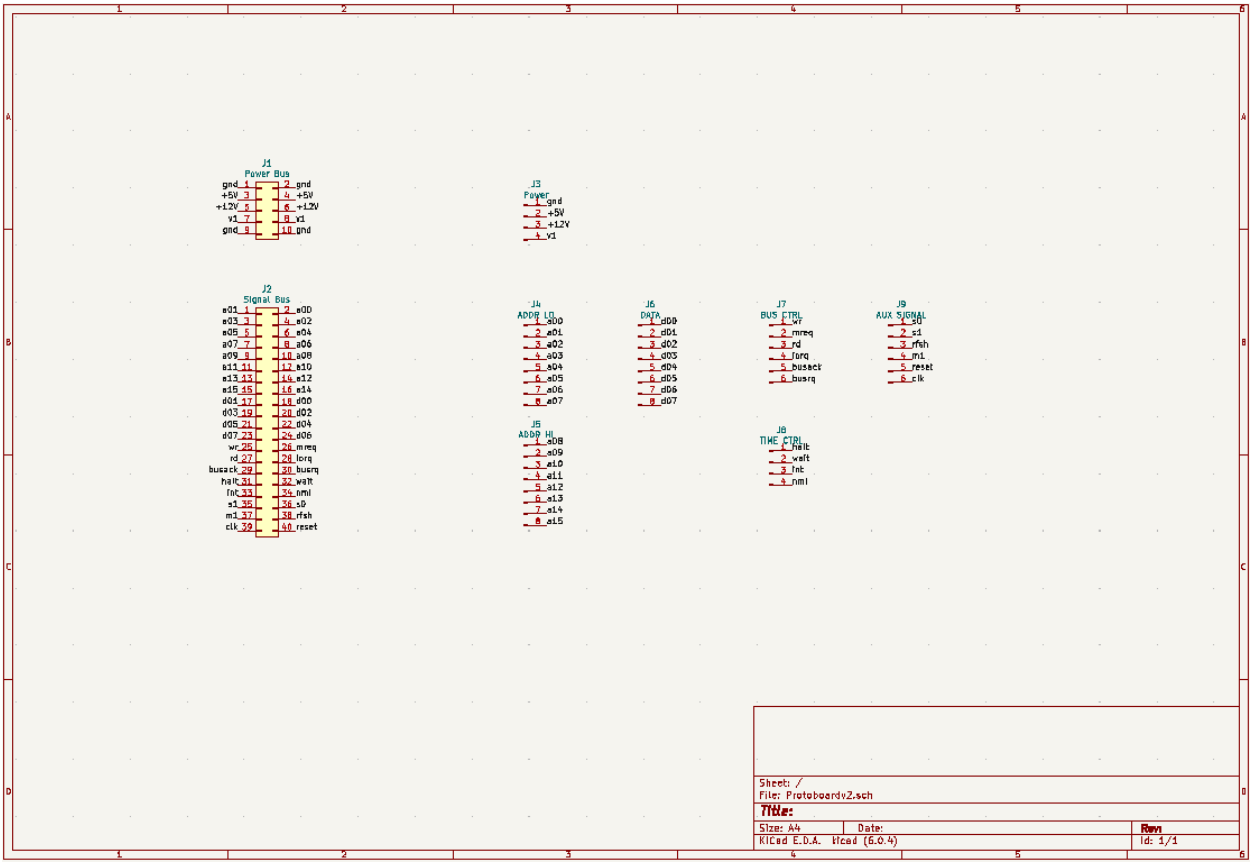
* only needed if assembled in the rack.

Protoboard

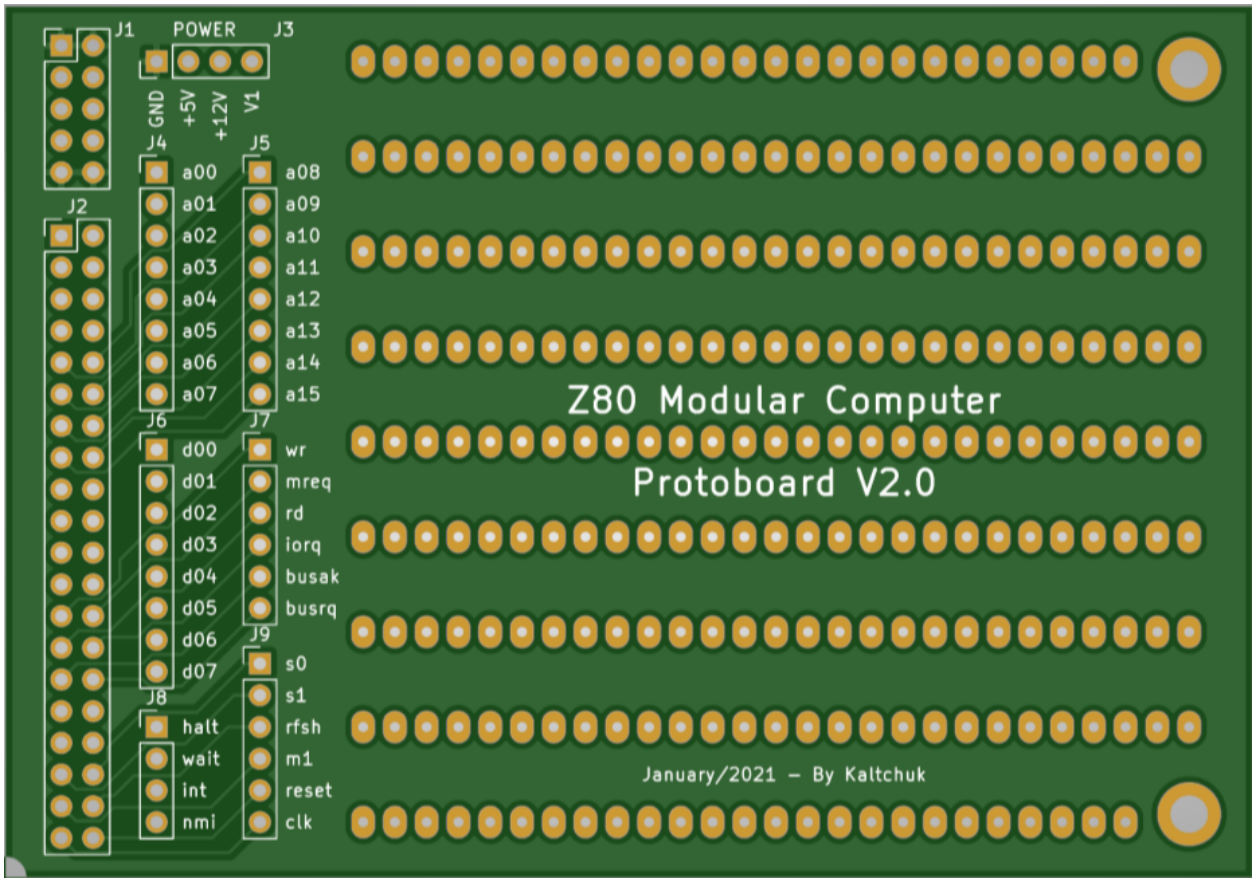
Description

The Protoboard is intended for experimentation and development of prototype modules for the Proton Z80MC. It was especially developed for prototyping using wire wrapping techniques.

All the signals from the signal and power buses can easily be accessed via the male pin headers.

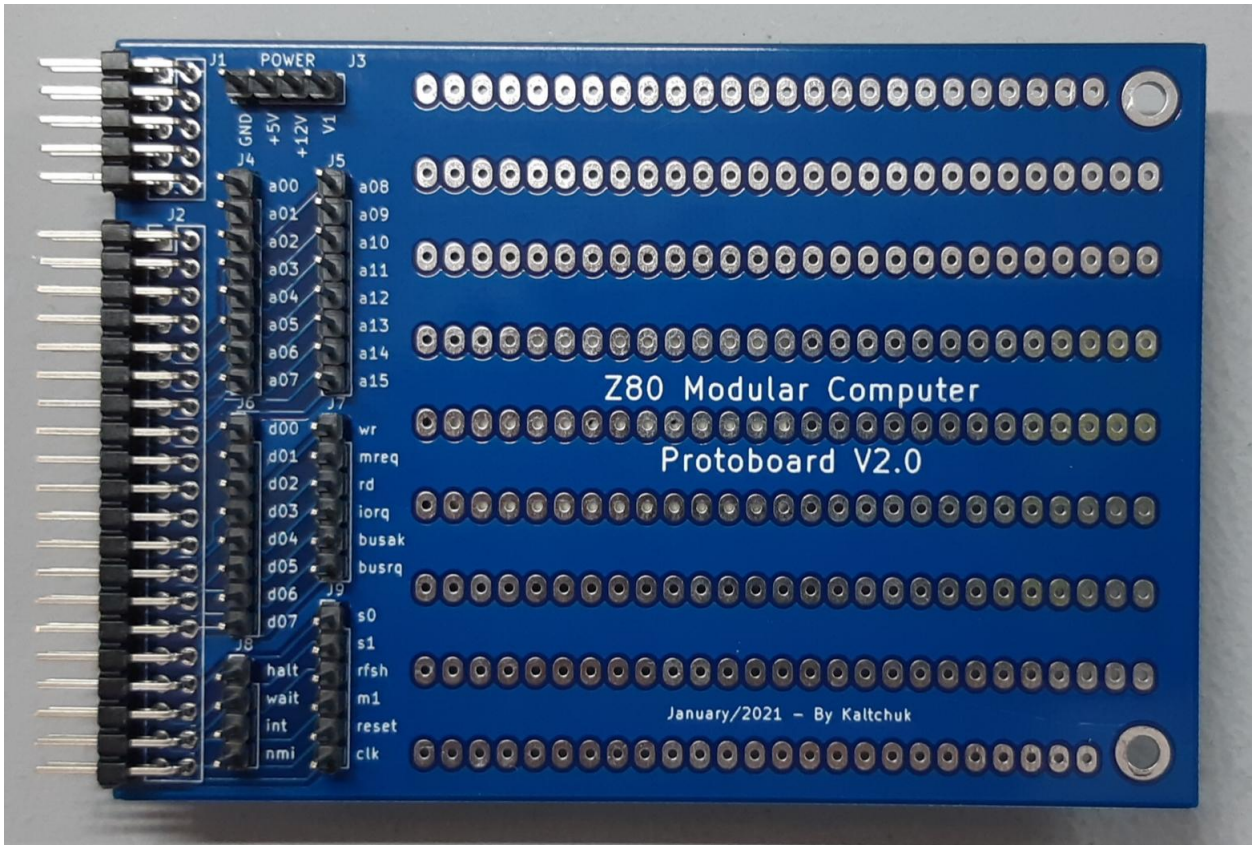


Protoboard schematic.



Protoboard PCB.

The picture below shows a protoboard with all connectors mounted, ready for prototyping. Note that connectors J3 though J9 have long pins (20mm), for wire wrapping.



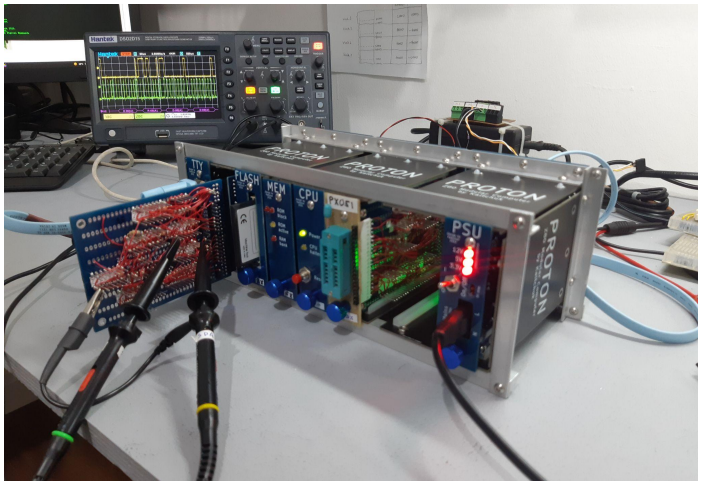
Bill of Material

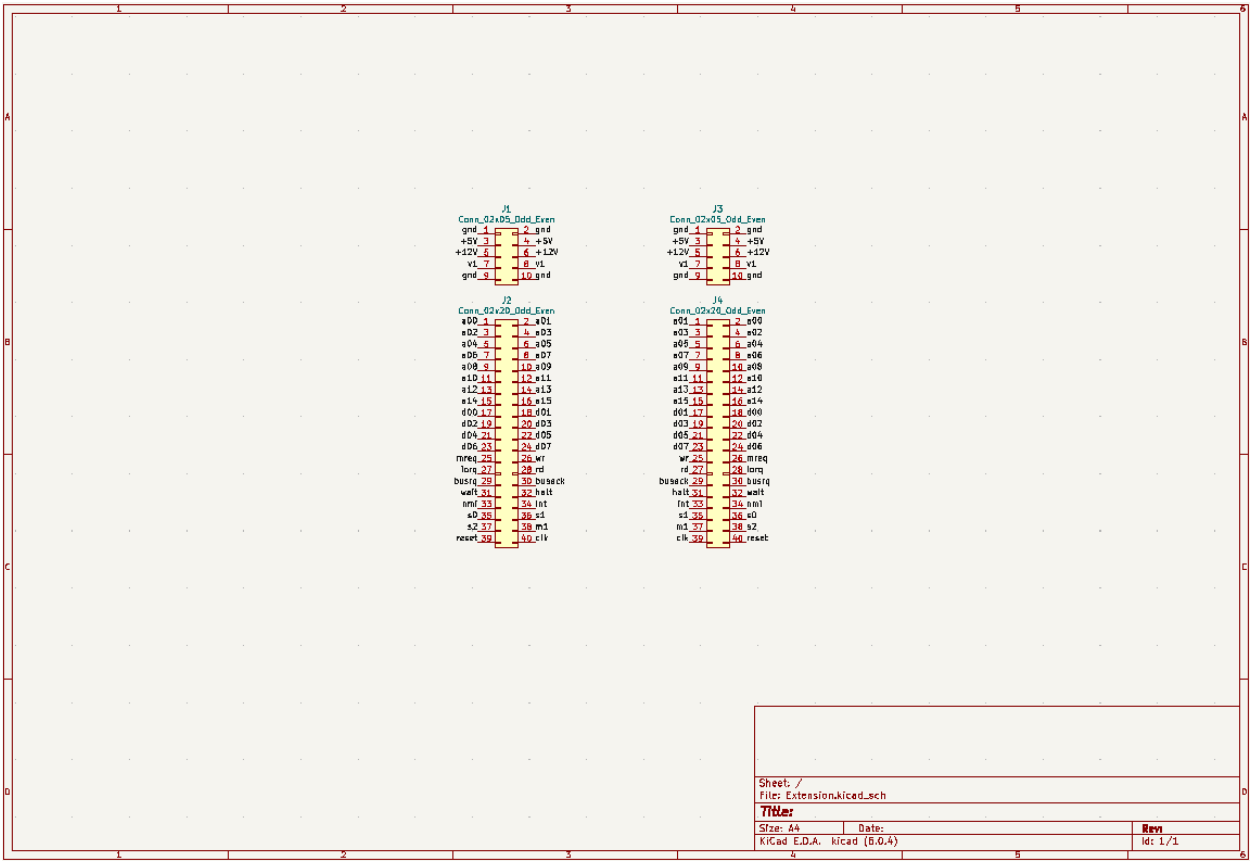
Item	Qty	Reference(s)	Value	LibPart
1	1	J1	Power Bus	Connector_Generic:Conn_02x05_Odd_Even_Male_90de
2	1	J2	Signal Bus	Connector_Generic:Conn_02x20_Odd_Even_Male_90de
3	1	J3	Power	Connector:Conn_01x04_Male_Long_Pins
4	1	J4	ADDR LO	Connector:Conn_01x08_Male_Long_Pins
5	1	J5	ADDR HI	Connector:Conn_01x08_Male_Long_Pins
6	1	J6	DATA	Connector:Conn_01x08_Male_Long_Pins
7	1	J7	BUS CTRL	Connector:Conn_01x06_Male_Long_Pins
8	1	J8	TIME CTRL	Connector:Conn_01x04_Male_Long_Pins
9	1	J9	AUX SIGNAL	Connector:Conn_01x06_Male_Long_Pins
10	1	PCB1	Protoboard PCB	

Extension Card

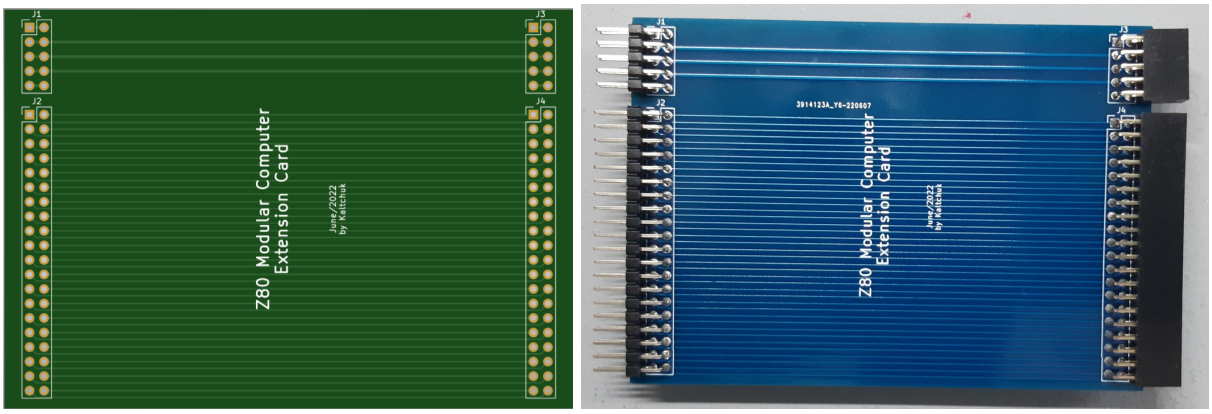
Description

Usually the development of a card requires troubleshooting with a multimeter, oscilloscope and logic analyzer. All these instruments require connecting probes to the circuit under analysis. If the card is inserted in the rack, this access can be very difficult or almost impossible. Due to this, I designed the Extension Card, so any card can be debugged “outside” the rack. This picture shows the I2C Card protoboard being debugged outside the rack, plugged to the Extension Card.





Extension Card schematic.



Extension Card PCB.

Bill of Material

Item	Qty	Reference(s)	Value	LibPart
1	5	J1	Conn_02x05_Odd_Even	Connector_Generic:Conn_02x05_Odd_Even_Male_90deg
2	5	J2	Conn_02x20_Odd_Even	Connector_Generic:Conn_02x20_Odd_Even_Male_90deg
3	5	J3	Conn_02x05_Odd_Even	Connector_Generic:Conn_02x05_Odd_Even_Female_90deg
4	5	J4	Conn_02x20_Odd_Even	Connector_Generic:Conn_02x20_Odd_Even_Female_90deg
5	1	PCB1	Extension PCB	

Top/Bottom/Lateral Card

Description

This card has no electrical function. It only has a mechanical function to connect to the backplane via the aluminum frames and hold the PCB card slots for the other cards. If you don’t intend to build the rack, this card is not necessary.

For a 9 slot rack, you’ll need a total of 8 of these cards - 3 on the top, 3 on the bottom and 2 on the laterals. You'll also need 18 PCB card slots. (Of course the lateral cards don’t need PCB card slots).

Bill of Material

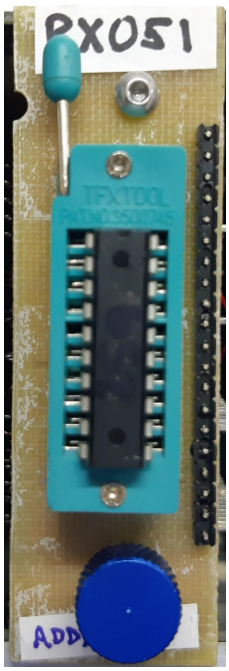
Item	Qty	Reference(s)	Value	LibPart
1	1	PCB1	Top/Bottom?Lateral PCB	
2	3	SLOT*	PCB slot	
3	4	Bolt**	3Mx5mm allen	
4	4	Nut**	3M	

* not needed for lateral mounting

** only needed if assembled in the rack.

PX051 Card (only in protoboard)

Description



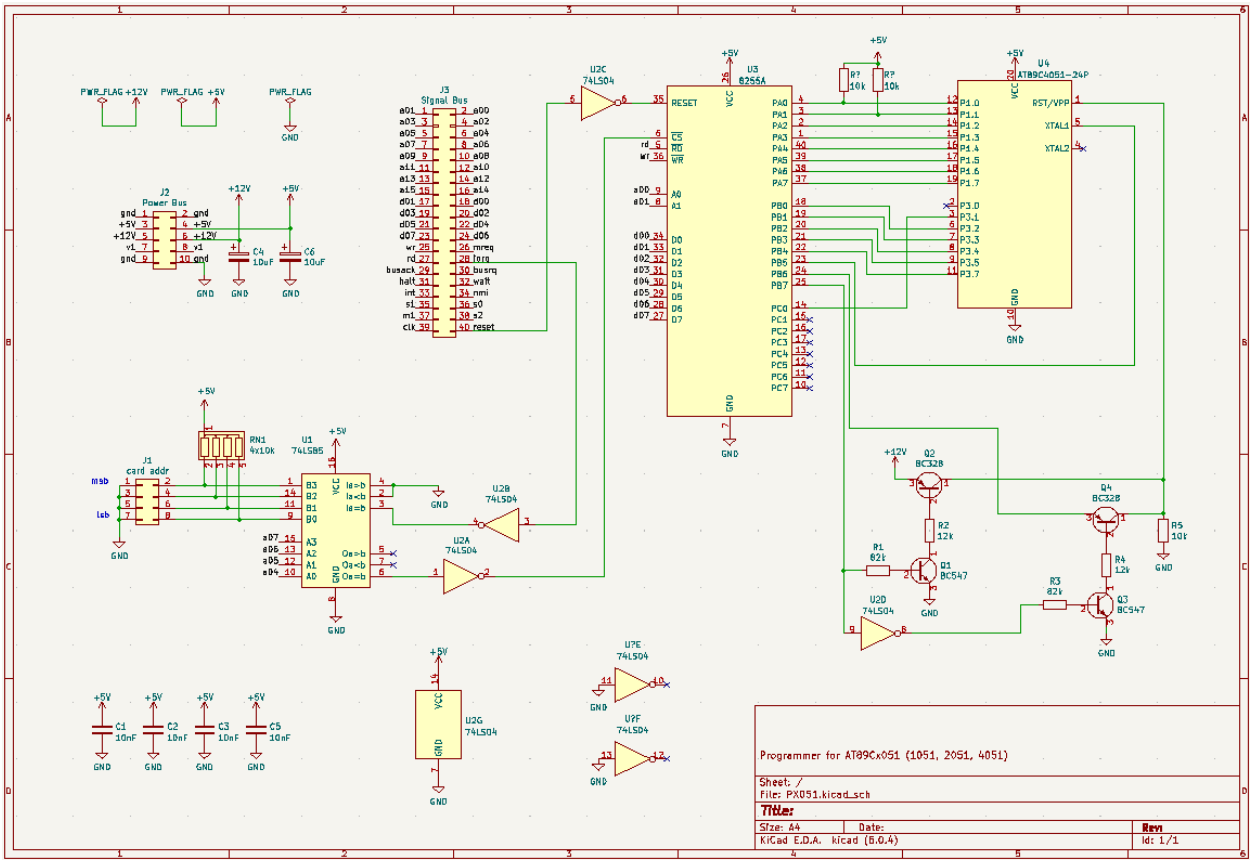
The PX051 Card is a programmer for the Atmel AT89Cx051 microcontroller family (AT89C1051, AT89C2051 and AT89C4051). It is used with the **PX.COM** utility in drive c:, which give the following options:

```
C>px
PX v1.0 by Kaltchuk, mar/2022.
AT89Cx051 programmer for PX051 card.

Use:    R       Read chip's memory.
        P       Program chip with binary file.
        L       Write lock bit.
        E       Erase chip.
        V       Verify if chip is erased.
        S       Read chip's signature.
        Q       Quit the program.
        ?       Show this help screen.

#
```

The binary file that will be programmed to the microcontroller must be on the same drive of **PX.COM**, i.e. drive c:. Usually the binary file has a **.BIN** extension.

Make sure that the PX051 Card is assigned to address D, otherwise the program won't work.

The frontal panel has a 20 pin narrow ZIF socket that holds the microcontroller being programmed. It doesn't necessarily have to be mounted on the frontal panel, but it definitely makes things easier. Another option would be to mount the ZIF socket directly on the protoboard and plug the PX051 card on an Extension Card so you can insert and remove a microcontroller on the ZIF socket without removing the PX051 Card from the rack. You can find the schematic and PCB layout of the frontal panel in the Github repository.



PX051 schematic.

Bill of Material

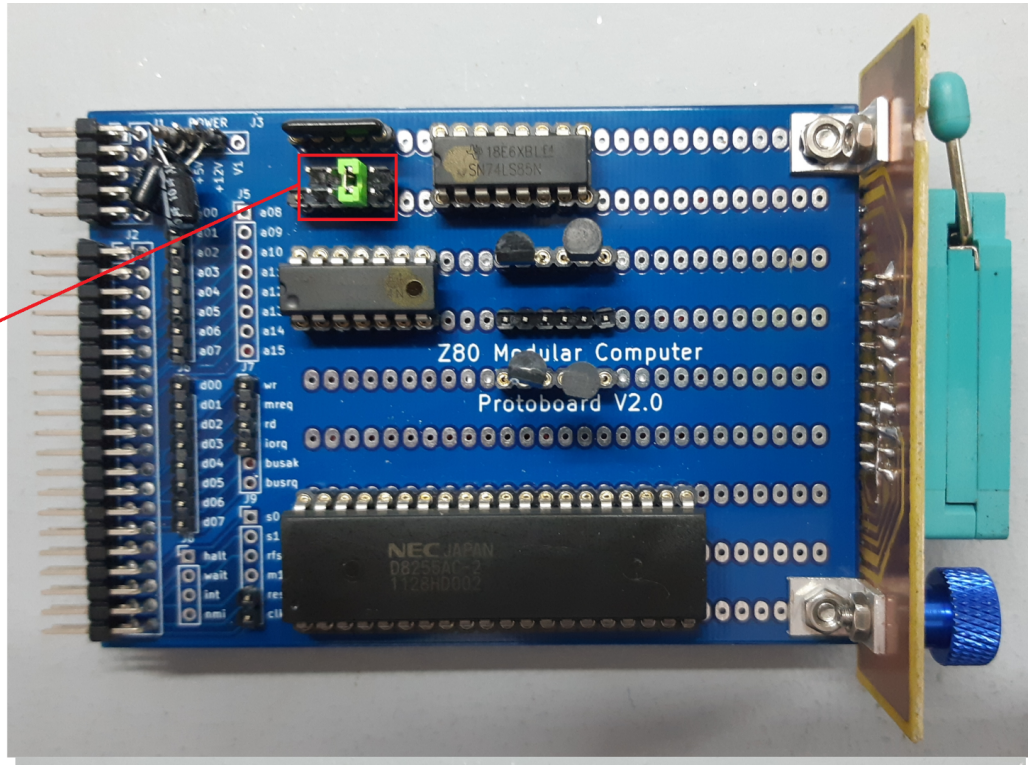
Item	Qty	Reference(s)	Value	LibPart
1	4	C1, C2, C3, C5	10nF	Device:C
2	2	C4, C6	10uF	Device:CP
3	1	J1	card addr	Connector_Generic:Conn_02x04_Odd_Even_Male
4	1	J2	Power Bus	Connector_Generic:Conn_02x05_Odd_Even_Male_90deg
5	1	J3	Signal Bus	Connector_Generic:Conn_02x20_Odd_Even_Male_90deg
6	2	Q1, Q3	BC547	Transistor_BJT:BC547
7	2	Q2, Q4	BC328	Transistor_BJT:BC328
8	2	R1, R3	82k	Device:R
9	2	R2, R4	12k	Device:R
10	1	R5	10k	Device:R

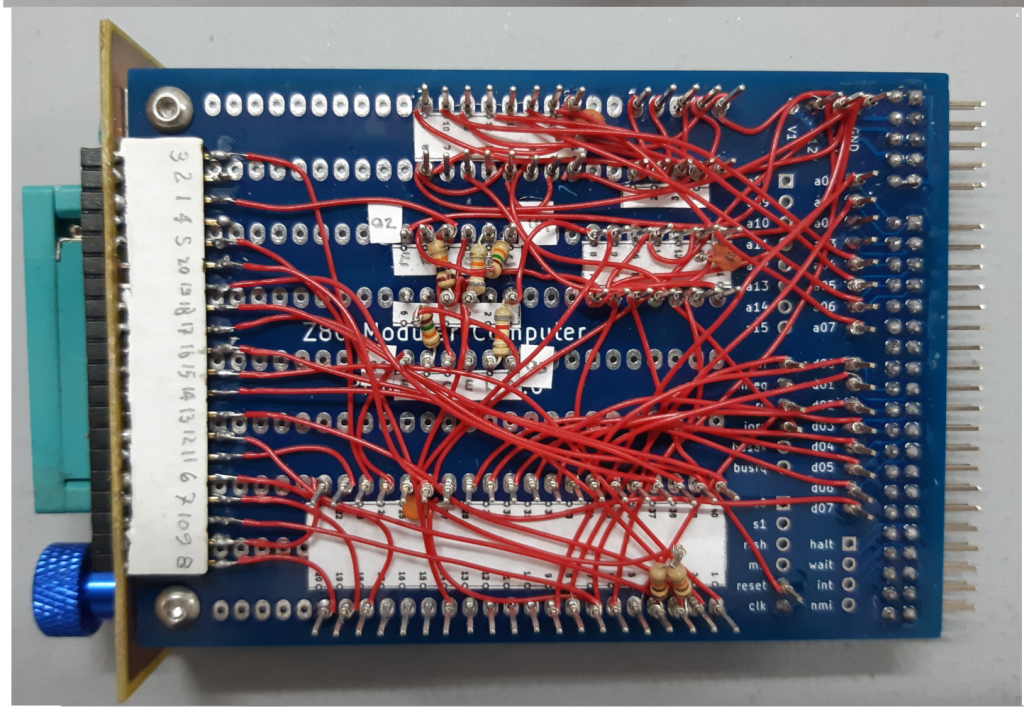
11	2	R6, R7	10k	Device:R
12	1	RN1	4x10k	Device:R_Network04
13	1	U1	74LS85	74xx:74LS85 + socket
14	1	U2	74LS04	74xx:74LS04 + socket
15	1	U3	8255A	Interface:8255A + socket
16	1	U4	AT89C4051-24P	20pin_narrow_ZIF
17	1	U5	74LS04	74xx:74LS04 + socket
18	1	PCB1	Prototype PCB	
19	1	PCB2	Px051 frontal PCB	
20	3	bolt	3Mx5mm allen	
21	1	knob	3Mx5mm	
22	4	nut	3M	
23	2	AL	10x10x1x8mm	aluminum profile

Configuration

Card address (J1). This is the address that the CPU will use to communicate with this card via IN/OUT instructions. It is a nibble from 0 to F. No jumper means 1 and a jumper on means 0. The picture shows the card configured with address D.

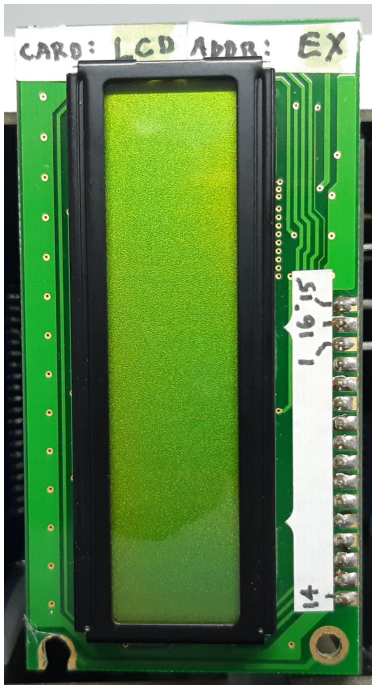
J1: card addr





LCD Card (only in protoboard)

Description



The LCD Card was initially developed as an easy output during the debugging phase of the USART Card. It can be used as a pseudo printer.

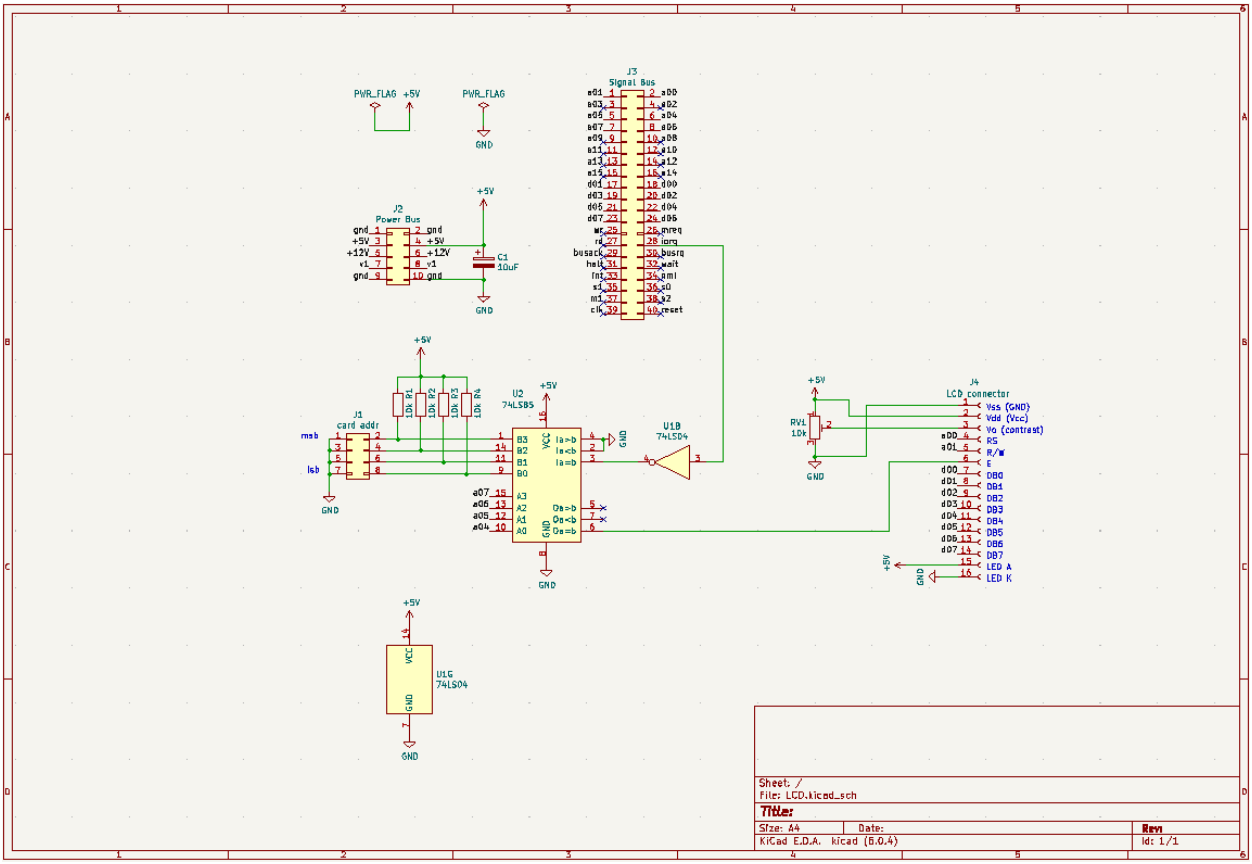
Before using this card, a driver has to be installed. The easiest way to do this is by using the **`LCDDRV.COM`** utility in drive A. This utility will install the LCD driver at the top of TAP memory (0CE0h) and modify the physical device jump table on the BIOS so that the LCD can be addressed as the CP/M standard printer (LPT:). The LCD Card must be configured with address E in order to use the driver configuration utility.

```
A>lcddrv
LCD driver installed @ 0CEC0.
LPT: => LCD card

A>
```

Now the LCD will show a blinking block, indicating that the LCD Card is ready to be used.

Notice that the LCD is actually the frontal of the LCD Card.

Trimpot RV1 is used to adjust the display contrast.



LCD Card schematic.

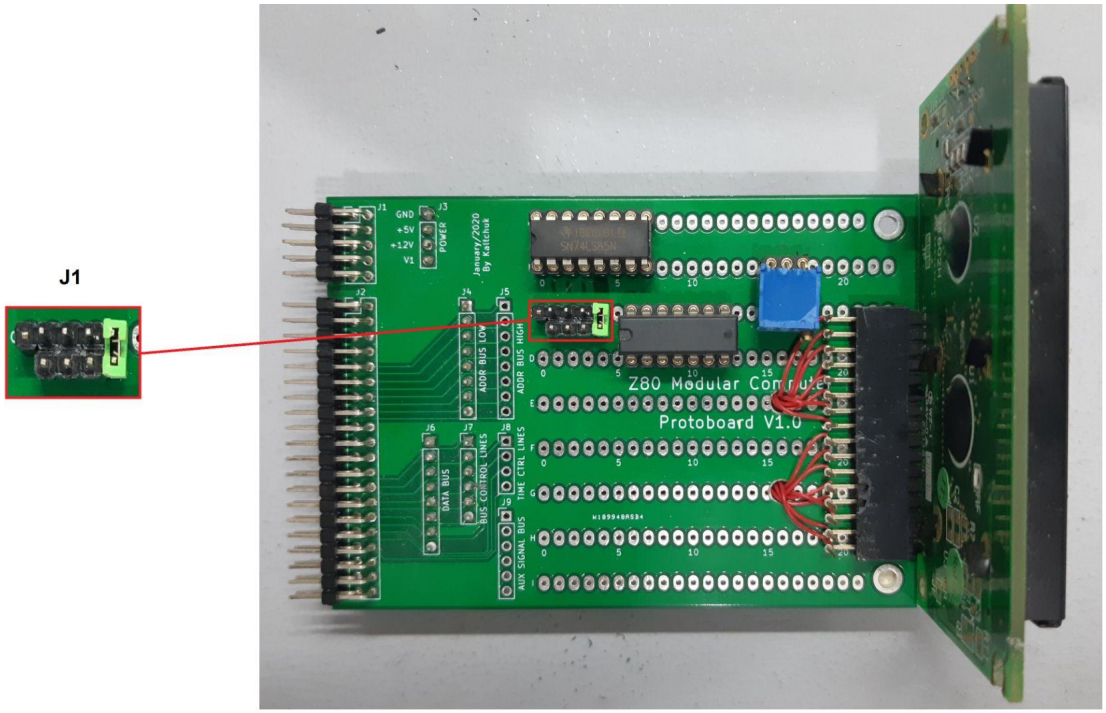
Bill of Material

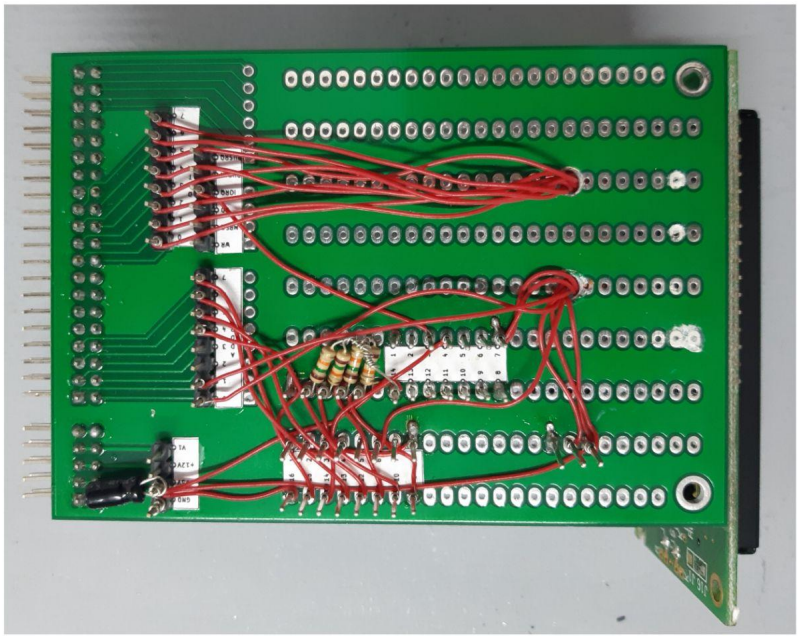
Item	Qty	Reference(s)	Value	LibPart
1	1	C1	10uF	Device:CP
2	1	J1	card addr	Connector_Generic:Conn_02x04_Odd_Even_Male
3	1	J2	Power Bus	Connector_Generic:Conn_02x05_Odd_Even_Male_90deg
4	1	J3	Signal Bus	Connector_Generic:Conn_02x20_Odd_Even_Male_90deg
5	1	J4	LCD connector	Connector:Conn_01x16_Female
6	4	R1, R2, R3, R4	10k	Device:R
7	1	RV1	10k	Device:R_POT_TRIM
8	1	U1	74LS04	74xx:74LS04 + socket
9	1	U2	74LS85	74xx:74LS85 + socket
10	1	LCD	16x2 LCD	

11	1	PCB1	Prototype PCB	

Configuration

Card address (J1). This is the address that the CPU will use to communicate with this card via IN/OUT instructions. It is a nibble from 0 to F. No jumper means 1 and a jumper on means 0. The picture shows the card configured with address E.





USART Card (obsolete - only in protoboard)

Description

This was the first attempt at serial communication, and was implemented via an Intel 8251 USART. It has only one serial port and a maximum baud rate of 38400bps. It uses an interrupt (Interrupt Mode 0) to signal the CPU that a byte has arrived. This interrupt generates a conflict with some softwares that uses address 0038h for interrupt service, like SID, ZSID and DDT.

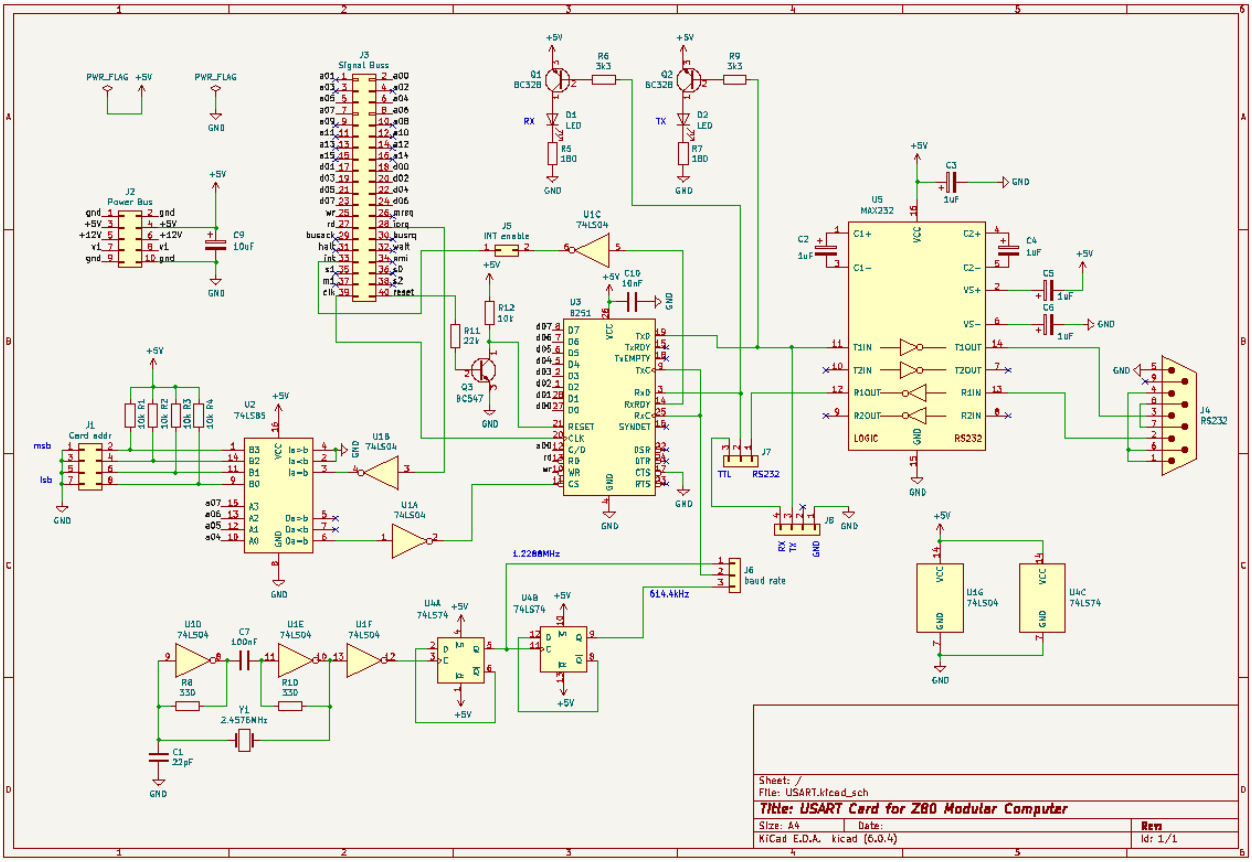
The baud rate is selected using a combination of hardware and software, i.e. jumper J6 and the baud rate factor on the mode instruction word, which is a prescaler for the clock signal (CLK). The USART Card has a clock circuit with a 2.4576MHz crystal. This clock signal is fed to two cascaded D-type flip-flops, resulting in 1.2288MHz and 614.4kHz. One of these frequencies is sent to the USART's clock pin via J6. The table below shows the possible combinations with different J6 and instruction word combinations.

J6	Baud Rate Factor (mode instruction word)	Baud Rate (bps)
614.4kHz	1	614400*
	16	38400
	64	9600
1.2288MHz	1	1228800*
	16	76800*
	64	19200

* impractical baud rate.

Afterall, with the 2.4576MHz crystal, only three baud rates are possible: 9600, 19200 and 38400bps. The card also has a 4 pin connector (J8) for TTL level serial communication.

Another important thing to notice is that CP/M and Monitor will only recognise the USART Card with BIOS prior to version 2.0.



USART Card schematic.

Bill of Material

Item	Qty	Reference(s)	Value	LibPart
1	1	C1	22pF	Device:C
2	5	C2, C3, C4, C5, C6	1uF	Device:C_Polarized
3	1	C7	100nF	Device:C
4	1	C9	10uF	Device:C_Polarized
5	1	C10	10nF	Device:C
6	2	D1, D2	LED	Device:LED
7	1	J1	Card addr	Connector_Generic:Conn_02x04_Odd_Even_Male
8	1	J2	Power Bus	Connector_Generic:Conn_02x05_Odd_Even_Male_90deg
9	1	J3	Signal Buss	Connector_Generic:Conn_02x20_Odd_Even_Male_90deg
10	1	J4	RS232	Connector:DB9_Male

11	1	J5	INT enable	Connector_Generic:Conn_02x01_Male
12	1	J6	baud rate	Connector_Generic:Conn_01x03_Male
13	1	J7	Conn_01x03	Connector_Generic:Conn_01x03_Male
14	1	J8	Conn_01x04	Connector_Generic:Conn_01x04_Male
15	2	Q1, Q2	BC328	Transistor_BJT:BC328
16	1	Q3	BC547	Transistor_BJT:BC547
17	5	R1, R2, R3, R4, R12	10k	Device:R
18	2	R5, R7	180	Device:R
19	2	R6, R9	3k3	Device:R
20	2	R8, R10	330	Device:R
21	1	R11	22k	Device:R
22	1	U1	74LS04	74xx:74LS04 + socket
23	1	U2	74LS85	74xx:74LS85 + socket
24	1	U3	8251	USART_8251:8251 + socket
25	1	U4	74LS74	74xx:74LS74 + socket
26	1	U5	MAX232	Interface_UART:MAX232
27	1	Y1	2.4576MHz	Device:Crystal
28	1	PCB1	Prototype PCB	

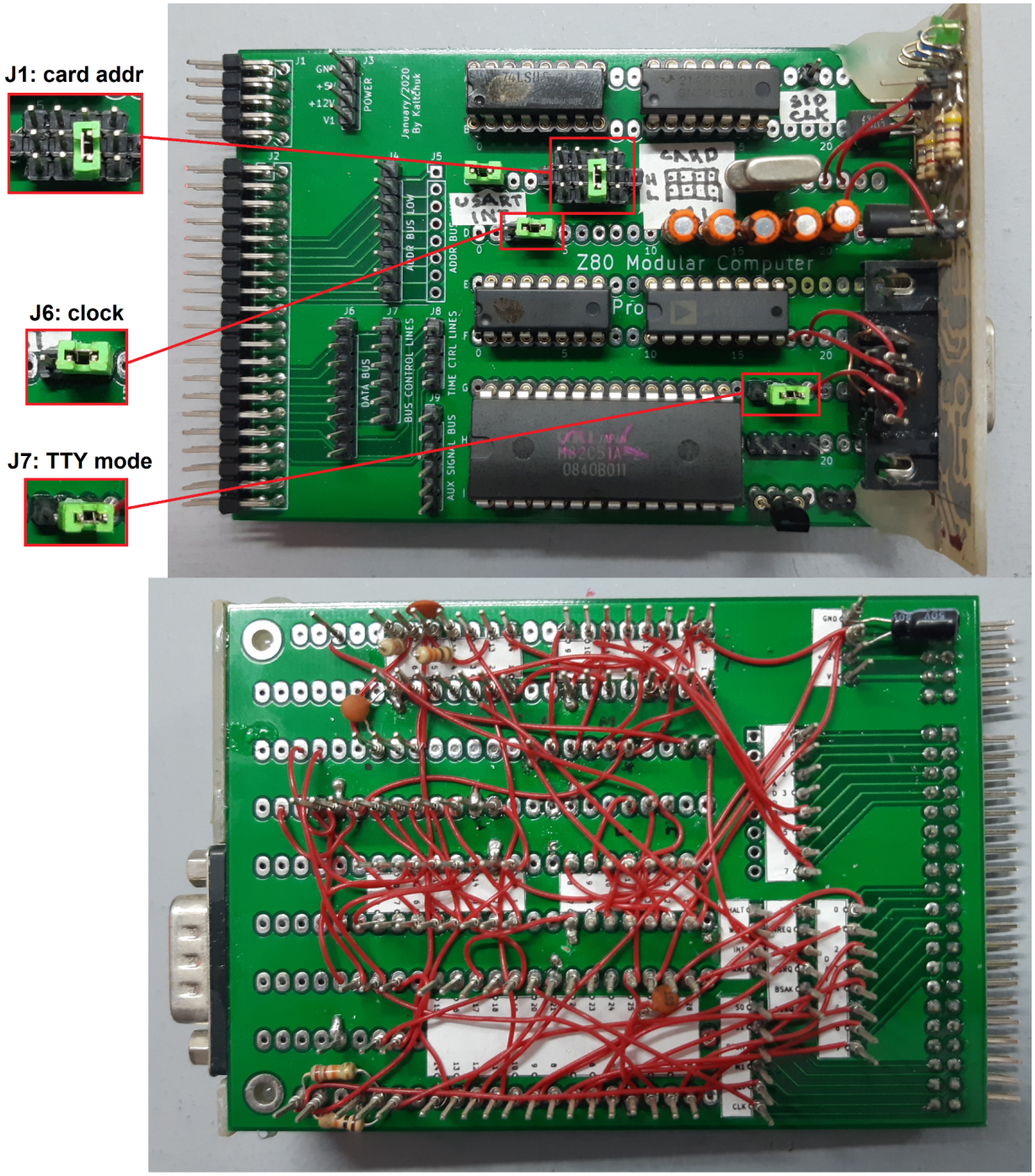
Configuration

There are three configurations needed on the USART Card.

Card address (J1). This is the address that the CPU will use to communicate with this card via IN/OUT instructions. It is a nibble from 0 to F. No jumper means 1 and a jumper on means 0. The picture shows the card configured with address D.

Clock (J6). This jumper selects the clock frequency for the USART's CLK pin. There are two options - 1.2288MHz or 614.4kHz. The picture shows 614.4kHz selected.

TTY mode (J7). This jumper selects if the port will communicate using RS232 or TTL level. The picture shows the port configured asRS232.


J1: card addr
J6: clock
J7: TTY mode
Z80 Modular Computer
Pro
USART INT
SIO clk
CARD
ADDR BUS LOW
DATA BUS
BUS CONTROL LINES
TIME CTRL LINES
AUX SIGNAL BUS
POWER
January/2020
By KatiChuk

Building Procedure

Follow these steps to build your own **Proton Z80 Modular Computer**:

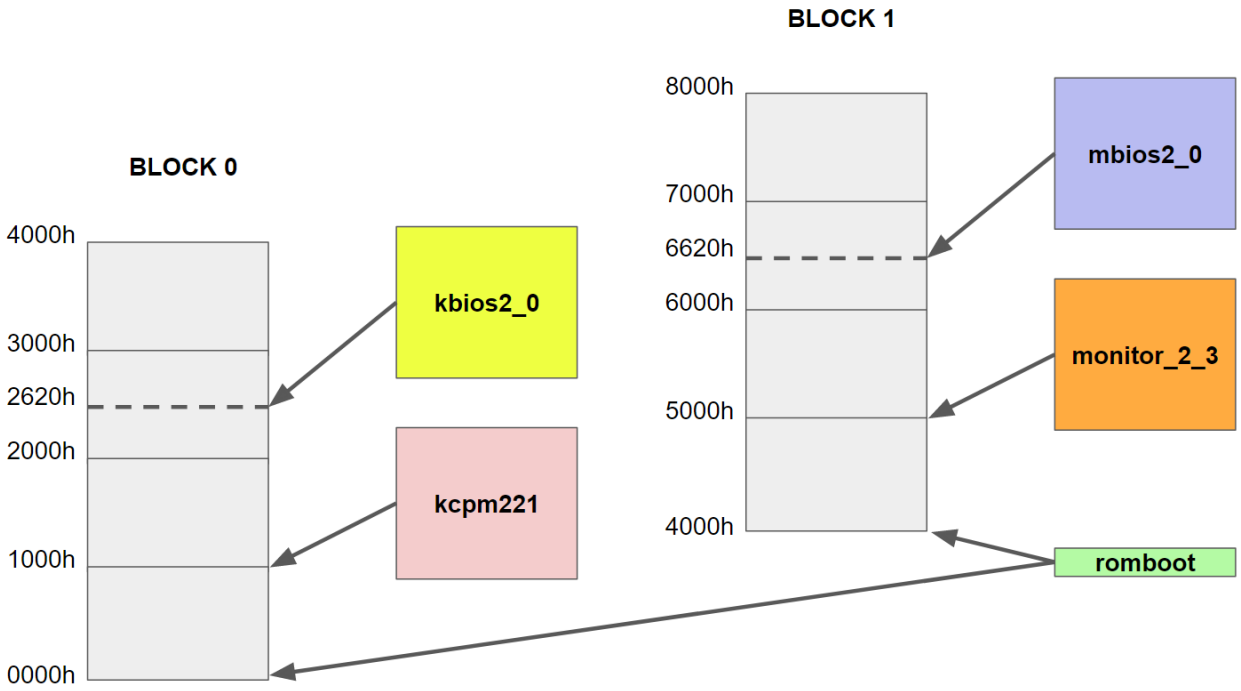
1. **PCB cards.** You can simply order the PCB cards from PCBway and JLC PCB. Each card has a directory inside the “KiCAD files” directory. If the card has a PCB design, you’ll find the gerber files and drill files. All these files are already inside a zip archive.

2. **Components.** Each card has its own bill of material.

3. **Cards Assembly.** After you gather all the PCBs and components, it's time to assemble the cards. Check the pictures to see what the cards should look like.

4. **Enclosure.** This is the most difficult part because there are no “final parts” for sale. You’ll have to cut and drill all the aluminum profiles. The enclosure is not essential to get the Proton running, but it will surely give a professional look & feel.

5. **EEPROM.** You have to burn the EEPROM according to the following memory topology:



kbios2_0.obj, kcpm221.obj and romboot.obj are here:
https://github.com/KALTCHUK/Z80_Modular_Computer/tree/master/CPM%20build

mbios2_0.obj, monitor_2_3.obj and romboot.obj are here: https://github.com/KALTCHUK/Z80_Modular_Computer/tree/master/Monitor%20build/Monitor%202

6. **Microcontrollers.** At least one of the ATmega328 on the TTY Card has to be programmed. Both, if you intend to use also the second port (TTY1). You can find the program under Atmel Studio directory (https://github.com/KALTCHUK/Z80_Modular_Computer/tree/master/Atmel%20Studio/preTTY/preTTY). I used Atmel Studio 7 with an AVRISP mkII programmer.

7. **Configuration.** Each card has a few jumpers that have to be configured. In the chapter for each card you'll find a section called "Configuration". It's important to notice that the address of each card is "hard coded" in the BIOS. So if you intend to use different addresses, you'll need to modify the BIOS.

   At this point, all the hardware is ready and configured. Insert all the cards in the rack, insert a 128MB compact flash in the FLASH Card, connect the power cord to the mains; and connect port TTY0 (DB9 connector) to your computer, via a USB-RS232 converter. (I suppose your computer doesn't have a native RS232 port). On your computer, open the serial terminal emulator and configure the port to 8N1, 250kbps. I strongly recommend the use of Teraterm as your serial terminal emulator. Select ROM block 1 (Monitor) and turn on the Proton. You should see the initial message from the TTY Card:

   ```
   *** TTY Card - firmware v1.1. ***
   *** by Kaltchuk, sep/2021.    ***
   ```

   After that, comes the boot message from Monitor:

   ```
   Z80 Modular Computer by Kaltchuk 2020.
   MBIOS 2.1
   Monitor by Kaltchuk 2020.
   ```

   Read the Monitor User's Manual and start exploring the Proton.

   Now, you can try to boot CP/M. Select ROM block 0 (CP/M) and turn on the Proton. You should see the same initial message from the TTY Card as before, and the boot message from CP/M:

   ```
   Z80 Modular Computer by Kaltchuk 2020.
   KBIOS 2.0 - 128MB Compact Flash.
   CP/M 2.2 Copyright 1979 (c) by Digital Research
   ```

8. **Disk drives.** In order to use the Flash Card emulating floppy disk drives on CP/M you have to format all 16 logical drives on the flash card. Use the Monitor for this task. Boot from Monitor (ROM block 1) and execute the **`format`** command for each one of the 16 drives (A, B, C… P).

   ```
   >format a
   Format disk A (y/n)? y
   Format complete.
   ```

   To check if everything went ok, boot from CP/M (ROM block 0) and use the **`DIR`** command to view the content of each drive.

   ```
   A>dir
   No file
   ```

   All 16 drives should be empty. Notice that the **`format`** command does not erase the previous content of the compact flash, it only cleans the FAT so CP/M will see an empty disk.

9. **Software.** So far, so good. Now it's time to download the software, but first, we need some tools. The first tool we need is **`XMODEM.COM`**. Boot from Monitor and type:

   ```
   >xmodem r 0100
   ```

   Send **`XMODEM.COM`**, which can be found here: https://github.com/KALTCHUK/Z80_Modular_Computer/tree/master/CPM%20software/Backups/tools (Of course, I'm assuming you already downloaded XMODEM.COM from Github to your computer before you attempt to send it to Proton with xmodem).

   After the transmission is complete, switch to ROM block 0 (CP/M), WITHOUT TURNING OFF THE PROTON. This procedure will swap from Monitor to CP/M without losing the file you just downloaded to the RAM. Press the reset button and type:

   ```
   A>save 4 xmodem.com
   ```

   Type **`xmodem`** and you should get this message:

   ```
   A>xmodem
   MISSING FILE NAME

   XMODEM 1.3 (10MHz build) - by Kaltchuk, 2021.
   Use: XMODEM [drive:]file_name -<option>
   options:    R to receive file
               S to send file
   ```

```
(Ctrl-X to abort transmission)
```

Now we have to download **NULU.COM**, which is the librarian, used to gather several files inside one .LBR file. This makes life much easier for up/download via xmodem. **NULU.COM** can be found in the same Github directory as **XMODEM.COM**. Type:

```
A>xmodem nulu.com -r
```

Send **NULU.COM**.

To test if **NULU.COM** downloaded correctly, type **nulu** and you should see this message:

```
A>nulu
NULU 1.52  (07/12/87)
Copyright (C) 1984, 1985 & 1987 by Martin Murray
Bug fixes in version 1.52 by Mick Waters

TYPE -H FOR HELP


NULU    .COM | XMODEM  .COM |
Drive A: Total 8160k, Used 20k, Free 8140k

-Open a library A0:>
```

Type **-x** to exit **NULU**.

At this point, you're ready to download all the software. As I mentioned before, the 128MB compact flash has 16 drives (a, b, c… p). Except for **XMODEM.COM** and **NULU.COM**, that you just downloaded to drive a:, all the rest is empty. The recommended content is shown in the table below:

```
            SUMMARY OF DISK CONTENT
            =======================

A:CP/M transient commands and utilities.
---------------------------------------------------------------
B:MBASIC v5.21 - Microsoft Basic Interpreter, and Basic programs.
  BASCOM v?.?? - Microsoft Basic Compiler.
---------------------------------------------------------------
C:BDS C Compiler v1.60 and C programs.
---------------------------------------------------------------
D:SLR Z80 Assembler v1.30 and .Z80 files.
---------------------------------------------------------------
```

```
E:WordStar 3.0
  TE 1.70 text editor
------------------------------------------------------------
F:FORTH v2.1.0
------------------------------------------------------------
G:Sorcim Supercalc v1.12 and spreadsheets
------------------------------------------------------------
H:dBASE II - Ashton-Tate Database Manager
------------------------------------------------------------
I:
------------------------------------------------------------
J:
------------------------------------------------------------
K:
------------------------------------------------------------
L:
------------------------------------------------------------
M:area used for backup and restore
------------------------------------------------------------
N:area used for backup and restore
------------------------------------------------------------
O:
------------------------------------------------------------
P:draft area
```

As you can see, only drives a: through h: are used. The easiest way to download all these programs is downloading the latest backup files from Github (https://github.com/KALTCHUK/Z80_Modular_Computer/tree/master/CPM%20software/Backups) with **XMODEM** and unpacking them with **NULU**. First, let's download all the backup files.

Suppose that the latest backup from drive a: is A220128.LBR, which was produced on 28/01/22. You have to download it to drive n:

```
A>xmodem n:a.lbr -r
```

Send A220128.LBR. Repeat this procedure for the other drives (b, c, d, e, f, g and h). After that, drive n: should look like this:

```
A>dir n:
N: A        LBR : B        LBR : C        LBR : D        LBR
N: E        LBR : F        LBR : G        LBR : H        LBR
```

Second, let's extract the content of each file and put it on the respective drive. The procedure for drive a: is a bit different. We'll extract the content to drive m: and copy from m: to a:, like this:

```
A>nulu -o n:a.lbr -e m:*.* -x
```

Go to drive m: and type:

```
M>pip a:=m:*.*
```

This command will copy all files from drive m: to drive a:.

Now go back to drive a: and type **`DIR.`** You should get something like this:

```
A>dir
A: NULU     COM : HALT     COM : PROTON   COM : DELBR    COM
A: STAT     COM : PIP      COM : NULU     DOC : DDT      COM
A: DUMP     COM : ED       COM : LOAD     COM : USQ      COM
A: SUBMIT   COM : ZSID     COM : SCREEN   COM : BACKUP   DOC
A: DIRALL   SUB : SQ       COM : UNERA    COM : LU       COM
A: UNARC    COM : ANYDISK  COM : XMODEM   COM : LDIR     COM
A: UNCRUNCH COM : XDIR     COM : CRUNCH   COM : UNZIP    COM
A: LCDDRV   COM : ARK      COM : D        COM : NULU     INF
A: PK       COM : PK       DOC : HP       COM : HP       DOC
A: N41      COM : N41      DOC : N41      ASM : BACKUP   SUB
A: TTYBAUD  COM : DISK     MAP : RELEASE  NOT : CLS      COM
```

Maybe the order is different, but the content should be very similar.

Repeat the extraction procedure for drives b: through h:. Don't forget to change the destination drive in the **`nulu`** command line (**`-e <dest_drive>:*.*`**). For these drives you can extract the files directly to the destination drives; no need to use the **`PIP`** command. For example, the procedure for drive b: would be like this:

```
A>nulu -o n:b.lbr -e b:*.* -x
```

Congratulations, mission accomplished! Feel free to explore your new **Proton Z80 Modular Computer** with **CP/M 2.2** and have fun!

Please, send me an email with your feedback (kaltchuk@gmail.com).