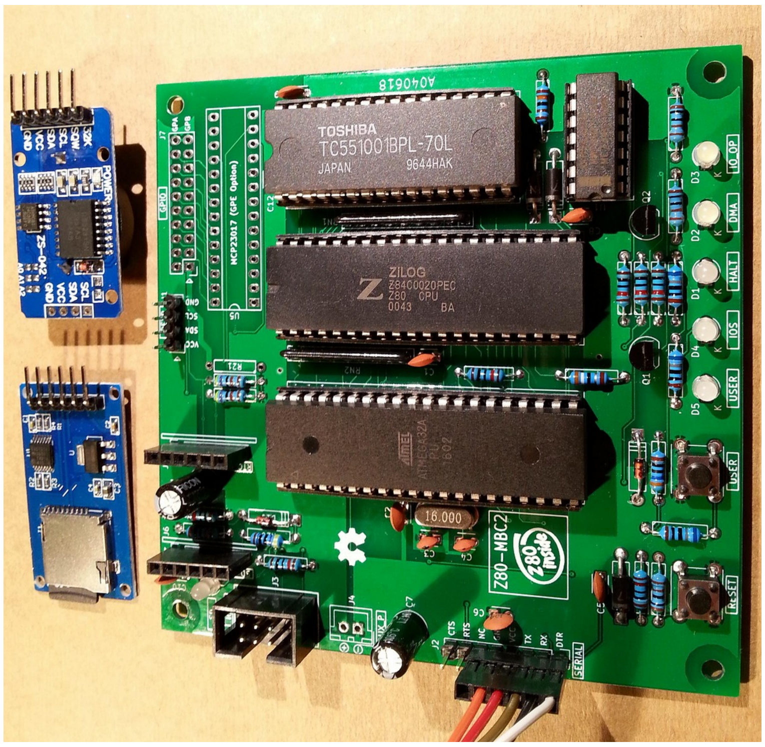
IT vintage equipment

Z80-MBC2 - a 4 ICs homebrew Z80 computer

Item No	
Name	**Z80-MBC2 - a 4 ICs homebrew Z80 computer**
Q'ty	1
Source	e-auction (PCB)
Date	2020/09/20-29
Comments	This document has been prepared for own use and therefore it follows a certain look and feel. It contains a combination of Internet-based knowledge and own experience. Don't get confused by words like 'I' or 'we' - sometimes it means the author of internet document, other times it reflects our own opinion. Internet sources are clearly marked and links are provided. Big THANKS to all contributors and authors of the remarkable Z80-MBC2 microcomputer! -DarS007 GitHub: https://github.com/DarS007/Z80-MBC2_guide

sources:
https://hackaday.io/project/159973-z80-mbc2-a-4-ics-homebrew-z80-computer from 2018;
https://github.com/SuperFabius/Z80-MBC2

Z80-MBC2 - Homemade 8MHz Z80 SBC, 128kB banked RAM, RTC, SD (HD emulation), Basic and Forth interpreters, CP/M 2.2 and 3, cross Assembler and C (SDCC)



1 ARCHITECTURE OVERVIEW

1.1 WARNING - Z80-MBC2 IS DIFFERENT !

Just a word of caution: **Z80-MBC2** is **NOT** a typical microcomputer you might know from 70's and 80's of the previous century. You will **NOT** find here a **glue logic** you might be familiar with. Instead, get ready for a new, genius concept of merging two worlds: **AVR/ATmega** and **Z80**.
ATmega plays a role of system supervisor that prepares a comfortable environment for it's Z80 mate. And only when everything is ready, Z80 is enabled to run and execute it's user code.
This mimics well the modern 'software-defined XXX' trend. Z80 clock, for instance, is generated by ATmega and it's value can be controlled by the software (4 or 8 MHz). The same with hard disks (emulated by ATmega and SD card) or serial interface (emulated by ATmega).

Be ready for the downsides of this 'software-defined' approach. You have to take care **NOT ONLY** of Z80 code, but also of the ATmega controller code. Without running ATmega controller, your Z80 microcomputer is dead.

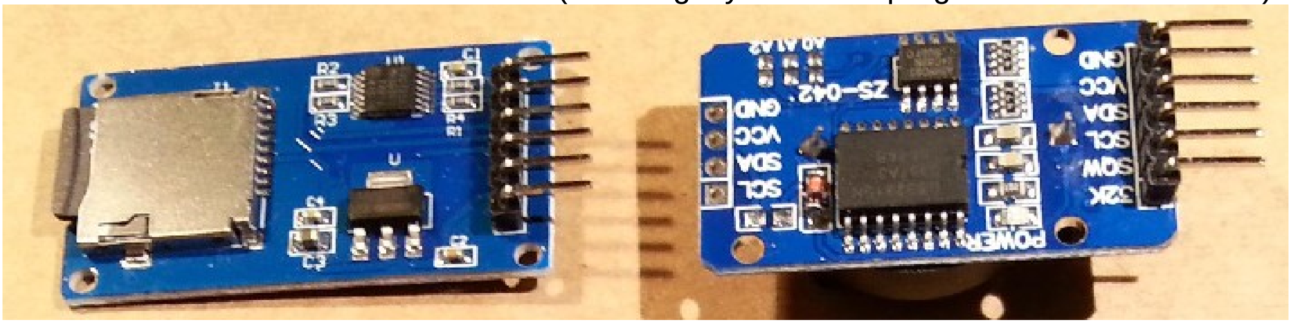
Tribute to Z80-MBC2 designers! They did a tremendous work of merging two worlds. Their genius approach resulted in a clean, minimalist hardware design (all the 'intelligence' is in the code now). Wow!

1.2 INTRODUCTION

source:
https://hackaday.io/project/159973-z80-mbc2-a-4-ics-homebrew-z80-computer

The Z80-MBC2 is an easy to build Z80 SBC (Single Board Computer). It is the "evolution" of the Z80-MBC (https://hackaday.io/project/19000), with a SD as "disk emulator" and with a 128KB banked RAM for CP/M 3 (but it also runs CP/M 2.2, QP/M 2.71, UCSD Pascal and Collapse OS too).

It has an optional on board 16x GPIO expander, and uses common cheap add-on modules for the SD and the RTC options. It has an "Arduino heart" using an Atmega32A as EEPROM and "universal" I/O emulator (so a "legacy" EPROM programmer is not needed).



Pict. SD module (optional for IOS-LITE, mandatory for IOS) and RTC module (optional).

It is a complete development "ecosystem", and using the iLoad boot mode it is possible cross-compile, load and execute on the target an Assembler or C program (using the SDCC compiler) with a single command (like in the Arduino IDE).
The needed ICs for the "base system" are:

- Z80 CPU CMOS (Z84C00) 8 MHz or greater
- Atmega32A
- TC551001-70 (128kB RAM)
- 74HC00
- MCP23017 - if you want the 16x GPIO expansion (GPE option)

1.3 Z80 PROCESSOR

The 74HC00 is used as RS flipflop to stop the Z80 CPU during I/O operation, giving the needed time to the Atmega32A to interact with the Z80 bus, and as part of the MMU.

Note that only the CMOS version of the Z80 CPU can be used here. Only CMOS version, under given condition that are respected in this schematic, has logical levels compatible with Atmega32A and 74HC00.

1.4 ATmega - EPROM, I/O SUBSYSTEM AND CLOCK

The MCU Atmega32A is a core element of the system (even more important than the base Z80 microprocessor). It is used as:

- universal I/O subsystem (including serial interface)
- Eeprom
- reset
- 4/8 MHz clock generator for the Z80 CPU.

At boot time, Atmega32A uses uTerm or uCom to communicate with user for selection of software to be run on Z80. Then Atmega32A configures the hardware for selected Z80 runtime. For BASIC, for example, the ROM image:

- is stored inside the Atmega32A (only for IOS-LITE)
- and is loaded to TC551001 RAM by the Atmega32A during the system boot.

For full IOS (not IOS-LITE), CP/M executables are kept on microSD card (they need to be flashed there first), and Atmega32A presents them as virtual disks to Z80.

From the user comments on the project page:
The Z80-[ATmega] AVR interaction schema uses a Wait-Bus_Hold sequence and was designed to be time-independent.
[...]
The way that IO worked on the V1 board is that accesses to IO space would trigger the flipflop that would put the Z80 into "wait" state, essentially halting it. Then the AVR would figure out what was going on, put stuff on or take stuff off of the bus, and un-flip the flipflop, allowing the Z80 instruction to continue. Essentially, a really SLOW memory access, with the stuff that normally happens in memory or IO hardware implemented in AVR firmware instead.
I haven't looked at the V2 at all (I ad assumed it worked the same, until your message), but there's no reason that the same scheme couldn't work for Z80 memory accesses as

well. The V1 seems simpler to me - it's easier (?) to make the RAM access be full-speed, but it takes more pins.

1.5 SERIAL PORT FOR TERMINAL/CONSOLE

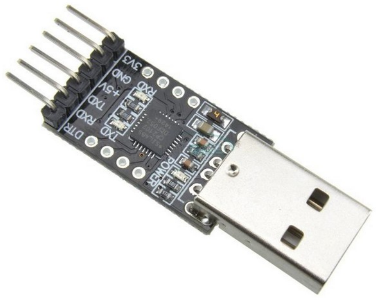
Serial connection is the main method of communication with Z80-MBC2 computer.
You can:

- A) use PC with external RS232(TTL)-USB adapter to connect to SERIAL port **J2** on the main PCB. You then don't need any other add-on boards like *uCom* or *uTerm*
- B) use ***uCom add-on board*** to connect PC or terminal hardware via:
  - standard RS232 interface (IC MAX232 and DB9 connector are on the uCom board), and/or
  - RS232(TTL)-USB adapter (it has a separate socket on the uCom board)
  - please note the both interfaces (standard RS232 and RS232(TTL)-USB adapter) can be connected at the same time. You can have your serial terminal connected to one interface (for user CLI), and your PC with Arduino IDE connected to the second (for firmware update), and flexibly switch between them.
- C) use ***uTerm add-on board*** to connect VGA monitor and PS/2 keyboard to form a standalone computer.
  Additionally, *uTerm* PCB also has a separate socket for RS232(TTL)-USB adapter, which can be used for connecting with PC.

Serial port speed

IOS-LITE	**9'600** bauds (bits per second)
IOS	**115'200** bauds
if you don't properly set fuse bytes (**H:D6, L:AF**) and the ATmega defaults to internal 1MHz clock	7'400 bauds (for IOS firmware)

Please note that several versions of RS232(TTL)-USB adapter exist. Make sure that pins of your adapter match the layout on Z80-MBC2 PCB.
DTR signal is used by Arduino IDE for Atmega "autoreset", therefore make sure your adapter has this signal too.



If your RS232(TTL)-USB adapter is designed to work with two voltage levels and it has 3.3-5 V jumper, set the jumper to **5V**.
Don't leave this jumper open! Otherwise your Z80-MBC2 will receive garbage due to the lack of power voltage for output stage of the adapter.

Our RS232(TTL)-USB adapter has the following pins:

DTR	**RXD**	**TXD**	**VCC**	CTS	**GND**

therefore a special 6-pin socket was prepared for soldering into the uCom board. It matches the 5-pin layout on uCom PCB and 'ignores' CTS signal (CTS pin was cut away and GND pin was bent).
This is how this 6-pin socket is seen from PCB side:

DTR	**RXD**	**TXD**	**VCC**	**GND**

1.6 FLASHING ATmega32A

Atmega32A needs to be programmed (flashed) with appropriate firmware (IOS or IOS-LITE) before Z80-MBC2 can run.
Two options exist:

- use Arduino IDE to compile the code and then to flash the Atmega32A via serial connection. This is thanks to the Atmego 32A bootloader. Or...
- use Arduino IDE only to compile the .ino code into HEX binary, and then use the external ISP programmer software to flash the Atmega32A with HEX via ISP interface

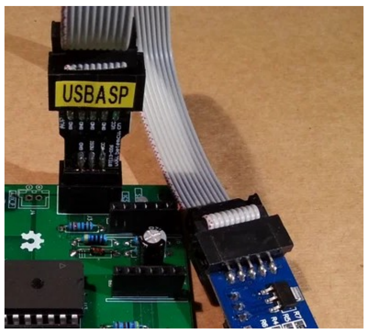
1.6.1 ISP INTERFACE HARDWARE

Read this interesting article:
How use the ICSP port with the USBasp programmer under linux to burn the bootloader
https://hackaday.io/project/159973-z80-mbc2-a-4-ics-homebrew-z80-computer/log/150087-how-use-the-icsp-port-with-the-usbasp-programmer-under-linux-to-burn-the-bootloader
for better understanding of ISP interface and it's role in flashing Atmega32A.

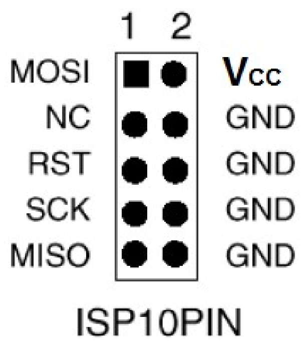
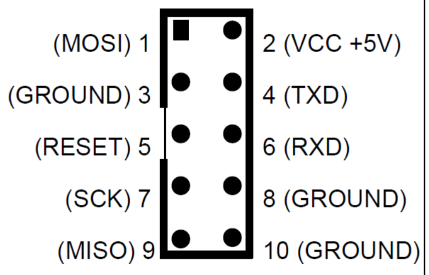
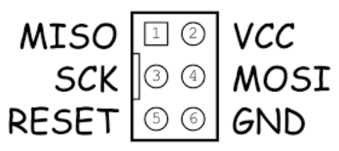
You can use the on board **ICSP port J3** (also called **ISP** port) to write the bootloader, but remember to disconnect any other connector when using it. Also both SD and RTC modules (if present) must be **removed** from the board when the ICSP port is in use.

USBasp programmer can burn the bootloader using Arduino IDE
(read further for more details).

You need to use a common 10pin-6pin adapter to connect the **USBasp** programmer hardware to the ICSP connector of the Z80-MBC2



Please note that the pinout of the USBasp is a little different from the "standard" ICSP (os ISP) pinout:

ISP 10 PIN (standard)	ISP 10 PIN (USBasp hw)	ISP 6 PIN (standard)
	 (MOSI) 1 2 (VCC +5V) (GROUND) 3 4 (TXD) (RESET) 5 6 (RXD) (SCK) 7 8 (GROUND) (MISO) 9 10 (GROUND)	

1.6.2 ‘ARDUINO IDE’ PLATFORM

If you want to fully rely on Arduino IDE, two steps are needed:

- First, flash the Arduino bootloader (with the method you prefer). MightyCore bootloader: https://github.com/MCUdude/MightyCore#how-to-install needs to be flashed to Atmega32A.
- Next use the Arduino IDE to compile and upload the IOS (or IOS-LITE) "sketch" to Atmega32A via serial interface.

Z80-MBC2 board meets the minimal setup circuit requirements for MightyCore bootloader (https://github.com/MCUdude/MightyCore#minimal-setup), therefore the bootloader works really well. It just functions in the background on ATmega, where it is waiting for appropriate serial transmission. Once a magic character is received, the bootloader activates itself and receives the firmware HEX file sent from Arduino IDE.
When the bootloader has been flashed to Atmega, you just connect the serial interface (standard RS232 from uCom or RS232(TTL)-USB adapter) to your PC, then select the correct serial **Port** under the **Tools** menu in Arduino IDE, and click the **Upload** button. Your code (compiled .ino sketch in form of HEX binary) will be transferred by Arduino IDE to Z80-MBC2 computer.

If you're getting some kind of timeout error, it means your RX and TX pins are swapped, or your auto reset circuity isn't working properly (the 100 nF capacitor on the reset line).

1.6.3 FLASHING Atmega BOOTLOADER

Sources:
1) An Easy to Build Real Homemade Computer: Z80-MBC2!
https://www.instructables.com/id/An-Easy-to-Build-Real-Homemade-Computer-Z80-MBC2/
2) How use the ICSP port with the USBasp programmer under linux to burn the bootloader
https://hackaday.io/project/159973-z80-mbc2-a-4-ics-homebrew-z80-computer/log/150087-how-use-the-icsp-port-with-the-usbasp-programmer-under-linux-to-burn-the-bootloader

"**MightyCore**" from https://github.com/MCUdude/MightyCore is the bootloader.

Remember to disconnect any other connector when using the ICSP. Also both SD and RTC modules (if present) must be removed from the board when the ICSP port is in use.

BURNING THE BOOTLOADER FROM ARDUINO IDE:
To easily burn the bootloader follow these "quick and dirty" steps (tested on a Linux Mint OS with Arduino IDE 1.8.5):

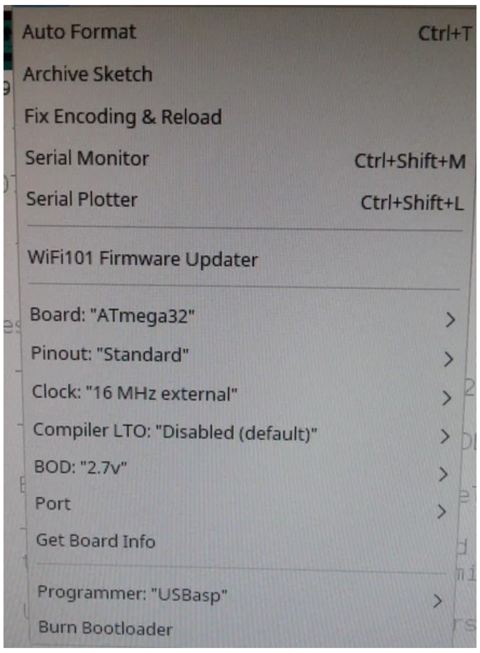
- **STEP 1:** Connect the 10 pins connector of the USBasp programmer to the 6 pins ICSP port (J3) of the Z80-MBC2
- **STEP 2:** Verify carefully that any other connector of the Z80-MBC2 is not used, and verify that both the SD and RTC modules (if present) are removed from the board;.
- **STEP 3:** Only at this point connect the USB side of the USBasp programmer to an USB port of your workstation;
- **STEP 4:** Open a "terminal" window on your workstation and go to the directory

where there are the Arduino IDE executables, and get the root privileges with the command:

```
sudo su
./arduino
```

- **STEP 5:** Because Arduino IDE is running as the root user it is necessary re-install the "core" for the Atmega32. Open the Board Manager as you already did. Note that you must do this step only the first time you execute the Arduino IDE as root;
- **STEP 6:** Now from the Tools menu of Arduino IDE select
  - "**Atmega32**" as "Board",
  - "**16 MHz external**" as "Clock",
  - and "**USBasp**" as "Programmer".
- Then you can burn the right bootloader (without playing with the FUSE setting) selecting "**Burn Bootloader**" from the same "Tools" menu.

Arduino IDE settings :



Board: "ATmega32"
Pinout: "Standard"
Clock: "16 MHz external"
Compiler LTO: "Disabled (default)"
BOD: "2.7v"
Port
Get Board Info
Programmer: "USBasp"
Burn Bootloader

1.6.4 ISP PROGRAMMER AND COMPILED HEX BINARY

If you do not want to use Arduino IDE and the bootloader, you can use an alternative method.
Use ISP software programmer (**AVRdude** PC software) and ISP hardware programmer (**USBasp**) to directly flash the compiled HEX binary to Atmega 32A.
Fuse bits:

High Byte	Low Byte	Lock Byte
0xD6	0xAF	0xCF

You can find the compiled HEX files on Z80-MBC2 web pages. However, please note that - for unknown reason - these files are available **only** for full IOS. Not for IOS-LITE.
Not a big deal though. You can always use the Arduino IDE to compile .ino file into HEX file, export the HEX file and then flash it to Atmega32A via ISP programmer (AVRdude + USBasp).

avrdude is one of the most popular ISP programmer software for Linux.

Read the content of Atmega32A flash memory and write to flash_read.bin file:
```
> sudo avrdude -p m32 -c usbasp -P usb -n -U flash:r:"flash_read.bin":r
```
This also reads the fuse settings, so you can check whether they need to be (re)set.

Write (flash) the new firmware from fw.hex compiled HEX binary file:
```
> sudo avrdude -p m32 -c usbasp -P usb -U flash:w:"fw.hex":i
```

Set the fuse/lock bytes
```
> sudo avrdude -p m32 -c usbasp -P usb -U hfuse:w:0xD6:m
> sudo avrdude -p m32 -c usbasp -P usb -U lfuse:w:0xAF:m
> sudo avrdude -p m32 -c usbasp -P usb -U lock:w:0xCF:m
```

Please note that the proper fuse setting **is very important**! You can often buy a second-hand Atmega32 chips (even if you think you're buying the brand new ones...) with randomly set fuse bytes. Our Atmega32, for instance, had the following:
```
avrdude: safemode: Fuses OK (E:FF, H:99, L:E1)
```
(internal oscillator 1MHz, boot flash section size 2k words)
When the proper HEX binary was flashed (but the fuses were left intact), the microprocessor was sooo slow. It output the serial data with mere 7'400 bauds (bits per second) instead of expected 115'000.
The only remedy was to flash the correct values for fuse bytes (`H:D6, L:AF`)
(external 16MHz, boot flash section size 256 words)
Btw, this seem to be a pretty common mistake. When you read the comments on:
https://hackaday.io/project/159973-z80-mbc2-a-4-ics-homebrew-z80-computer/discussion-121188 you can spot such entry:
Basically, the IOS light blinks very slowly (every 2 seconds) and there is serial output (either after a plain reset or User+Reset) but not to a baud rate minicom can handle. The SD LED also flashes around the same time serial output occurs. Probing with a scope I noticed that the Crystal wasn't oscillating, I removed the crystal and caps and I still get exactly the same behavior, it powers up, IOS light blinks slowly and serial output occurs but at a non-standard baud rate. There is also no activity on the Z80 side (ie. reset, D0-D7 are all idle).
This is exactly the result of flashing the firmware but **NOT SETTING** the fuse bytes.

You can check the on-line fuse calculator for Atmega 32:
http://eleccelerator.com/fusecalc/fusecalc.php?chip=atmega32

2 SOFTWARE

Both **IOS-LITE** and **IOS** are available on the project web pages.
The first one is a simplified version that doesn't support the SD.
The second one is a full featured version that **requires the SD module** (e. g. to run CP/M).

2.1 IOS-LITE AND IOS

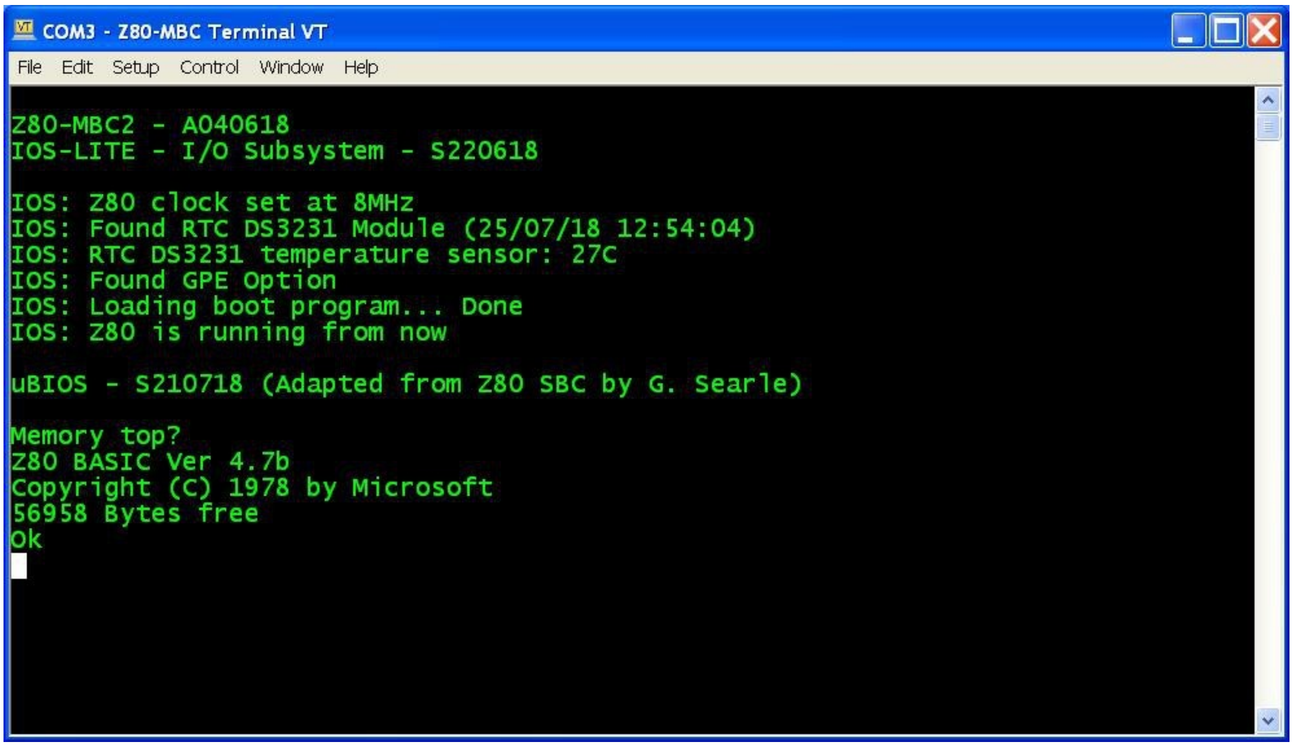
The current revision of IOS-LITE offers stand-alone versions of Basic and Forth.
Full-size IOS complements it with additional **CP/M 2.2, CP/M 3.0, QP/M 2.71, UCSD Pascal** and **Collapse OS** with **16 virtual disks** (8Mbytes each) for each OS.

Files on https://hackaday.io/project/159973-z80-mbc2-a-4-ics-homebrew-z80-computer web site:

FILE	CONTENT
IOS-LITE	
S220618_IOS-LITE-Z80-MBC2.ino	The "sketch" for the IOS-LITE. IOS-LITE doesn't support the SD (for full SD support use IOS S220718). x-arduino - 153.28 kB - 07/25/2018 at 16:13
IOS	
SD-S220718-R240620-v1.zip	The content of the **microSD** needed to run CP/M 2.2, CP/M 3.0, QP/M 2.71, UCSD Pascal and Collapse OS with IOS S220718-R240620. Adds Collapse OS. Zip Archive - 3.69 MB - 06/25/2020 at 10:41
S220718-R240620_IOS-Z80-MBC2.zip	The sketch for the **IOS** (with the needed libraries). Unzip into a folder and open the .ino file (with Arduino IDE). IOS must be uploaded into the Atmega32A flash. Adds support for Collapse OS. Zip Archive - 37.34 kB - 06/25/2020 at 10:40
S220718-R240620_IOS-Z80-MBC2.ino.with_bootloader_atmega32_16000000L.hex	The sketch for the **IOS in executable format** (.HEX) with the bootloader. This executable file is intended for use with ISP programmer like Atmel Ice or AVRISPmkII. **Fuse bits:** High Byte 0xD6, Low Byte 0xAF, Lock Byte 0xCF x-hex - 55.76 kB - 06/25/2020 at 10:39
APPLICATIONS FOR BASIC	
STARTREKV2.BAS	The famous game Super Startrek kindly debugged by a RetroBrew Computer Forum user. Play with Caps-Lock activated! Now the animation works… bas - 20.18 kB - 08/07/2018 at 16:32
TREKINST.BAS	Instructions for STARTREKV2.BAS bas - 6.80 kB - 08/04/2018 at 21:48
Examples with Basic.txt	Some examples showing the new "Virtual I/O Engine" with the stand-alone 8K Basic interpreter. plain - 4.07 kB - 07/31/2018 at 20:57

2.2 BASIC

From the project web pages:
I've "ported" the Basic interpreter to the Z80-MBC using the sources provided in the great Grant Searle site http://searle.hostei.com/grant/index.html, after the modification due the different HW design.
Grant requested an acknowledgment to his site to be included in the source code, so I did here (and I have also emailed to him about this thing).
The resulting ROM image is stored inside the Atmega32A (only for IOS-LITE) and loaded in the TC551001 RAM by the Atmega32A during the system boot. The original manual of this Basic interpreter is here:
http://www.nascomhomepage.com/pdf/Basic.pdf


COM3 - Z80-MBC Terminal VT
File Edit Setup Control Window Help

```
Z80-MBC2 - A040618
IOS-LITE - I/O Subsystem - S220618

IOS: Z80 clock set at 8MHz
IOS: Found RTC DS3231 Module (25/07/18 12:54:04)
IOS: RTC DS3231 temperature sensor: 27C
IOS: Found GPE Option
IOS: Loading boot program... Done
IOS: Z80 is running from now

uBIOS - S210718 (Adapted from Z80 SBC by G. Searle)

Memory top?
Z80 BASIC Ver 4.7b
Copyright (C) 1978 by Microsoft
56958 Bytes free
Ok
```

2.3 CP/M

According to the project page, CP/M 2.2 is running on the Z80-MBC2!
https://hackaday.io/project/159973-z80-mbc2-a-4-ics-homebrew-z80-computer/log/152560-cpm-22-up-and-running-on-the-z80-mbc2

CP/M 3 up and running on the Z80-MBC2!
https://hackaday.io/project/159973-z80-mbc2-a-4-ics-homebrew-z80-computer/log/154314-cpm-3-up-and-running-on-the-z80-mbc2

PREVIOUS VERSIONS
A new OS: QP/M 2.71 and a RTC for timestamping
https://hackaday.io/project/19000-a-4-4ics-z80-homemade-computer-on-breadboard/log/60037-a-new-os-qpm-271-and-a-rtc-for-timestamping

Misc stuff for Z80-MBC2
https://github.com/SupraJames/z80-mbc2

`memtest.asm / memtest.hex`	This program tests the memory of the Z80-MBC2
`monitor.asm / monitor.hex`	This is a simple Z80 Monitor program
`sketch-lcd`	for I2C attached LCD to display some debug info

3 HINTS AND FAQ

Q. I use the hex-file S220718-R280819_IOS-Z80-MBC2... on the Atmega 32 and after starting the Z80-MBC2 i got the boot-menu. So far so good! I explored the menu-entries and after choosing Forth, I run into Forth on every reboot. How can I get back to the boot-menu at startup?
A. Easiest way is to press both RESET & USER keys, release the RESET key holding the USER key down until the IOS led starts to blink, or you see the menu on the screen.

If you press and release the RESET key, D11 must do a single very fast blink (also if the SD module is not present) just before the IOS SD error is "printed" on the serial port. If you don't see it there must be an assembling error/fault.

Hard disk emulation
https://www.instructables.com/id/An-Easy-to-Build-Real-Homemade-Computer-Z80-MBC2/
Hard Disks are emulated using a microSD FAT16 or FAT32 formatted (a 1GB microSD is enough), so it is easy exchange the files with your PC (16 HDs for every OS are supported) using **`cpmtoolsGUI`** .

Q. How can I put some programs into the disk-images on the sd-card? Is there a tool to fill the disk-images on a PC? Under Windows?
A. If you scroll down on the 'Logs' menu you'll find **`CPMtools GUI`** and instructions. on how to use it. (You can of course use XMODEM to transfer files over to disk directly.)

Overclocking the Z80-MBC2...
https://hackaday.io/project/159973-z80-mbc2-a-4-ics-homebrew-z80-computer/log/155472-overclocking-the-z80-mbc2

Z80-MBC - OLDER VERSION
https://github.com/SuperFabius/Z80-MBC