Porting Altair BASIC to the IMSAI 8080

By Bruce E. Hall, W8BH

Altair Basic, written in 1975 by Bill Gates, Paul Allen, and Monte Davidoff, led to the birth of Microsoft. Bill Gates published the source code fifty years later in April 2025. It thought it would be cool to port this epic and historic piece of code to the IMSAI 8080. Read on if you are interested in experiencing "the birth of Microsoft" on your own IMSAI.

How to put Altair BASIC on your IMSAI 8080

Early on, Altair BASIC was supplied on paper tape. To use it you keyed in a short bootloader program on your computer. The program read the tape and started BASIC. Later systems included a monitor in ROM to load the tape. Even better, you could put BASIC on ROM, obviating the need for tape.

I created two versions of Altair BASIC. Here is an overview of installing BASIC as a ROM:

1. Download the ROM image file from GitHub.
2. Copy the ROM image file to the ROMS folder on your IMSAI's SD card
3. Edit the system.conf file, adding a memory map for 4K Altair BASIC at D800.
4. Set your NVS configuration to use the new memory map.

The next section goes into greater detail.

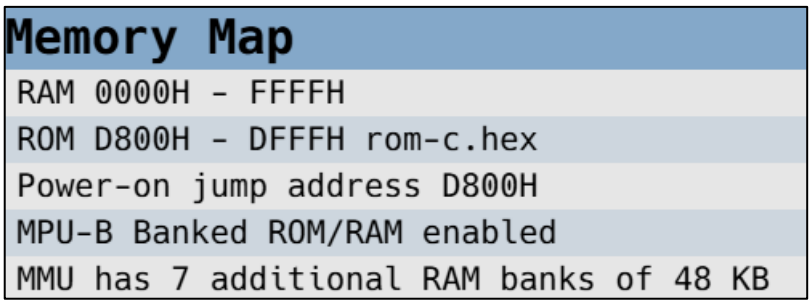
How to change your ROM

The ROM files are stored on the SD card in the imsai/roms directory. You may study the SD card on your PC and transfer files to and from this folder. You may also put as many ROM files in this folder as you like.

- *Download the basic-rom.hex file from GitHub and copy it into the imsai/roms directory on your SD card.*

Once the SD card is returned to the IMSAI, you can assign ROMS to 7 different memory maps, with zero or more ROMs getting assigned to each map. This assignment is done through the system.conf text file, located on the SD card in the imsai/conf directory. Confused?

All of that is best illustrated with an example. Use the IMSAI web interface to see your current Memory Map. Click on the SYS icon. My system displays the information shown here.



There is 64K of RAM, and 8K of ROM. The ROM, named "rom-c.hex", is located in memory at D800-DFFF. On <Pwr On>, the system will jump to D800 to begin execution. Banked ROM means that this ROM can be disabled to expose the full 64K of RAM. Without ROM banking, the 8K of RAM at D800-DFFF is effectively "hidden" by the overlying ROM and cannot be used. Banked RAM means that additional RAM can be swapped in and out of the 64K address space. See my article on bank switching for more information.

Changing the ROM is a two-step process:

1. Assign the ROM(s) to a memory map.
2. Change your system configuration to use the memory map.

To see the list of memory maps, go to the SYS window again and click on the gear icon in the top right corner. Now click on "system.conf" and scroll down to the list of memory configurations. The first map, MEMORY 1, may look like this:

```
[MEMORY 1]
# Default memory configuration used in most situations with MPU-A ROM:
# 256 pages RAM, 8 pages ROM (replaces or overlays RAM)
ram         0,256
rom         0xd8,8,mpu-a-rom.hex
# Start address of the boot ROM
boot        0xd800
```

The numbers to the right of RAM and ROM refer to pages of memory, where a page is 256 bytes. The two numbers for RAM indicate the starting point (page 0, or 0000h) and its length in pages (256 pages, or 64K). The entry for ROM, highlighted above, indicates that ROM file mpu-a-rom.hex is located on page D8 (D800h) and is 8 pages, or 2K in length.

Memory maps can be added, edited, or deleted. Be careful: an incorrect setting can render your system unresponsive. I recommend making a backup of this file (and the

whole SD card, for that matter) before making changes. My system came with 4 preconfigured memory maps.

Memory Map	Default Use
1	IMSAI monitor (booting CP/M)
2	IMSAI Monitor + VIO firmware
3	Memon/80 Monitor
4	XYBasic

To install Altair BASIC, we are going to add a memory map for it. Add the following lines to your system.conf file (after removing any existing memory map 5):

```
[MEMORY 5]
# for ALTAIR BASIC, use the following:
ram           0,256
rom           0xd8,16,basic-rom.hex
boot          0xd800
```

This map indicates that we are installing a 4K ROM at D800, and we will start execution at D800. Be sure to click on the Save (disk) icon before exiting.

After adding the memory map, you must tell the IMSAI to use it. This is done though the NVS Startup procedure, outlined here and here. Bits 8-10 of the address lines specify which memory configuration is used. For example, to use memory map #5 with the Auto-Run, Banked ROM, 4 MHz, Z80, and WiFi options, specify map **8D58**:

Auto-Run	Res	Res	Res	Banked ROM	Map	Map	Map	Unlimited	4 Mhz	Undoc Instr.	Z80	WiFi STA	Log Level	Log Level	POST
1	0	0	0	**1**	**1**	0	**1**	0	**1**	0	**1**	**1**	0	0	0
8				**D**				**5**				**8**			

Notice that the 'D' in '8D58' specifies use of Memory Map #5 (bits 101=decimal 5). You are not limited to this configuration. 0Dxx, 05xx, 8Dxx, and 85xx should all work.

That it! Altair BASIC is now installed. I hope you enjoy the experience as much as I have. I put a copy of the manual on my GitHub site. A small demo program follows. Continue reading if you'd like to know more.

Demo Program

Here is a small demo program, sine.bas, to show that Altair BASIC is working:

```
10 REM SCROLLING SINE WAVE
20 FOR X=0 TO 6.28 STEP 0.21
30 PRINT TAB(30*SIN(X)+31);"8"
40 NEXT X
50 GOTO 20
```

Copy it from your computer, then paste it into the TTY window with Ctrl-Shift-V. Type RUN to start the demo and press Ctrl-C to stop it.

Loading Altair BASIC via paper tape

Paper tape is not as convenient as ROM, but there's nothing like it for that retro experience! Besides, the emulated tape device on the IMSAI 8080 esp is just plain fun. I use it routinely to transfer software to and from my IMSAI.

You cannot load Altair BASIC from CP/M. You need to run a monitor program to use the paper tape version. I like the Memon/80 monitor in memory map #3. Or, if you are feeling adventurous, try the custom monitor program described in my ROM article.

To use Memon80, change your NVS configuration setting to 03xx or 83xx. I typically use 8358. You will be greeted by the "Memon/80 2.6 by M. Eberhard" prompt. Then,

1. Enter the "HL" tape reader command. You will not get any monitor reply.
2. Open the paper tape reader "PTR:" device.
3. Click on the PTR plug icon to connect it, turning the icon from red to green.
4. Drag the basic-tape.hex file from your PC and drop in the PTR device. At the "Load...?" prompt, click OK. If you want, you can also drop the file into your "TAPE:" library for future use.
5. Click 'Read' on the PTR device to start the tape. *Pro Tip: Click 'Read' again, up to 5 times, to make the tape go faster.*
6. Dots will appear on the TTY screen as the tape is read. When finished, the monitor will reply with the number of records read. Expect to see a number around 009C.
7. BASIC is loaded into memory. To run it, type "EX 0".

Porting Alair BASIC to the IMSAI 8080.

Altair BASIC was written, of course, for the Altair 8800, arguably the first commercially successful personal computer. The IMSAI 8080 debuted one year later and was made to be "plug compatible" with all the circuit boards of the Altair. Both computers used the Intel 8080 microprocessor.

The job of disassembling and converting machine code into human-readable 8080 assembly language has been already done. A wonderful explanation of the inner workings of the code is provided by Reuben Harris at altairbasic.org. Highly recommended. Charles Mangin cleaned up the code and posted it his GitHub site in 2019.

Earlier this year, Bill Gate's published the (almost complete) 3.0 version source code for Altair BASIC, referring to it as "the coolest code I've ever written." I recommend reading his description in Gates Notes. It contains a link to the downloadable code.

After a few hours and a few edits, I was able to re-assemble Charles Mangin's 3.2 version listing on my assembler of choice, Telemark Assembler (TASM). It was easier than I thought. But the code did not run. A blank screen stared back at me.

The first job was to find which routines involved input/output. Back in 1975, I/O was a serial port attached to a console. It did not take long to find the four I/O routines, which are:

a. Initialize the serial port(s)
b. Read a character
c. Write a character
d. Look for break (Ctrl-C) input

These tiny routines had to be rewritten so that they sent/received data over the correct I/O port.

I also discovered a typographical error in the online disassembled code. It was a small thing but hard to find. It is present in the code that assigns values to variables.

My biggest impediment was using a Z80 machine, or more accurately, the Z80 setting on my machine. Since the Z80 instruction set is a super-set of the 8080's, I could not imagine that would be an issue.

But it was. There are minor incompatibilities between the two processors. One incompatibility is the parity flag. Parity was a quick way (actually, at the time, the only quick way) to find errors in a stream of incoming serial data. On the 8080, the parity flag is set when the data has an even number of '1' bits. The Z80, on the other hand, also uses

this flag to indicate overflow conditions. Since parity checks are generally only used during serial data transfer, accounting for this incompatibility is a relatively easy chore.

Not so in Altair BABIC, I learned. Bill Gates and Co. used every trick they could to squeeze BASIC into 4K of memory. One trick uses the parity flag to decide if one number is greater than, equal to, or less than another number. Unfortunately, the Z80 gives a different answer than the 8080, depending on the numbers involved. This trick saved 2 bytes of code. Every byte was precious.

If you are curious about the parity trick, check out the two "JPO" instructions in the floating point out (FOut) routine. They cause a lot of havoc on a Z80 machine, since this routine is called to print numbers (including program line numbers).

The final hurdle was to create a bootloader. In the ALTAIR world, BASIC was loaded into the computer via paper tape. It cannot directly run from ROM because it contains many internal variables. It also relies on self-modifying code, another byte-saving trick. A bootloader is needed to move it from ROM to RAM.

I added a ton of comments after porting the code from Charles Mangin's site. Some of the comments come from altairbasic.org, some come from Bill Gates' original code, and some are my own. The resulting source file contains 2311 lines of code, 782 comment lines, and 233 blank lines, for a total line count of 3316. Check it out if you are curious about the inner workings of BASIC. You won't be disappointed going down that rabbit hole.

As a teenager with a TRS-80, I spent hours poking BASIC memory locations - dreaming that I would understand its code someday. Level I BASIC is not the same as Altair BASIC but I am checking it off my bucket list, nonetheless.

Happy Hacking!

Bruce.

Last updated: 22 Aug 2025

Other articles in this series: IMSAI 8080 articles