MiSTer Manual

Last Revised 2-Apr-22

Guide to MiSTer FPGA Computer and Console Cores

"Helpful instructions as to what each system's keyboard and controllers looked like and basic instructions and commands to use each system (and play its games)"

The MiSTer is an FPGA (Field Programmable Gate Array) based device that can simulate various computing devices. Picking the word simulate over emulate seems to create debate, but I will not delve any further into that other than to say that the big difference between this and having an emulator in your computer is that MiSTer can offer a more accurate, hardware (logic gate) based simulation of the original hardware.

The MiSTer's base hardware is a Terasic DE-10 Nano board and then there are several optional boards that can be added on. I think most would agree that the I/O Board is essential as is an SDRAM board, but there are several other interesting add-ons such as controller adapters, a real time clock, the ADC board (for using original cassette based software) and the MT32-Pi Hat. With MiSTer there's A LOT to explore and my hope is to get you started on that journey with your favorite systems that you had (or perhaps wished you had) back in the day.

Before jumping in it is important to know that a core is basically the piece of software that provides MiSTer with the logic gate level instructions as to how to simulate the integrated circuits in a given piece of hardware. Cores are controlled via the OSD (On Screen Display) generally asccessed via pressing F12 on your keyboard and that's the interface that lets you virtually insert disks, tapes and cartridges into your simulated device as well as adjust a myriad of other, core specific, settings such as perhaps adding memory, a hard drive, a new BIOS ROM…etc. It is also important to know that the MiSTer I/O board itself has 3 buttons:

1) Reset
2) On Screen Display (OSD)
3) Core Specific (Often used as a Core Reset button)

The purpose of the guide is to provide some basic info as to how to better use each device that MiSTer has a core for. Often I found myself floundering about trying to figure out how load a program on one of the computer cores or why a given video game console didn't do what I was hoping. This information is all available somewhere on the internet, but the goal is to consolidate it all and provide a starting point in a single guide that you can keep handily with your MiSTer device. This can be a slippery slope as to how much to include.

The goal of this guide is not to:

a) Help with core set up which may include obtaining some firmware (a ROM BIOS image) as that should all be part of the core's most basic documentation.
b) Finding software.

The goal of this guide is to:

a) Describe and show what the system physically looked like and how some of the core's simulated peripherals worked. For example, for all of the computer cores the first thing I provide is a picture of the keyboard. Often simulation means a button is a button so a PC keyboard used with MiSTer isn't going to be an exact match for System X's keyboard. There will be some mapping where this key on a PC keyboard translates to that key on System X's keyboard. Some systems added 'auto type' keywords that were printed on their keyboard while others had shortcuts to graphic symbols and I think it'd be nice to know how to use those. Some game consoles had odd (by today's standards) controllers, keypads and even keyboards. What controls (keyboard, gamepad, mouse…) would one have sitting in front of them had they just gotten this system on Christmas day and were opening it for the first time…

b) Systems from the early 80s often came with the BASIC computing language built in and much of what one would do when not playing games centered around learning BASIC. If the system supported a disk drive then usually there was some sort of DOS (Disk Operating System) to store files on the disk. On such system I am hoping this guide can provide a shortcut to learning how to type in, save, load and run a BASIC program such as

```
10 FOR X = 1 TO 10
20 PRINT "HELLO"
30 NEXT X
```

(which prints "HELLO" on the screen ten times) as well as how to load machine language programs from various supported media types. On later systems where GUIs made everything easier I'd like to document how to get set up from scratch. Often there are pre-loaded hard disk images available and that's GREAT… I love the convenience and the hard work put into creating those too. I'm sure many will be satisfied with that alone. However, I'd like to document that Christmas Day feeling for those that would like to experience setting up System X from scratch and also have cores that are true replacements for the original hardware such that I could feel comfortable doing new software development on them.

c) Provide a quick reference to the more obscure settings and the most common commands used with a given core along with links to user's guides and such for those that want to dig a bit deeper into System X. The idea that the next great game could be developed entirely by someone on a MiSTer is pretty exciting to me as is the idea that, despite old hardware parts gradually going bad, these devices will be preserved for future generations to not only see at a museum, but also to be able to experience and use at home.

Useful links (and sources of some of the included info):

https://github.com/MiSTer-devel/Main_MiSTer/wiki - MiSTer Wiki Home Page
https://github.com/theypsilon/Update_All_MiSTer - MiSTer Update scripts (highly recommended)
https://www.misterfpga.org – MiSTer FPGA Forum (lots of info and early release versions of cores)
https://github.com/Grabulosaure - MiSTer Intellivision Core (and more)

Credits (aka some helpful info contained within also came from):

https://pastebin.com/pM1XMe5E - MiSTer Computer Cheat Sheet by Owlnonymous

This guide is written from a North American, Windows user's perspective when it comes to preparing virtual hard drive images and such. If you would like to add anything to help Linux/OS X users and/or provide any more detailed system info please email feedback / update suggestions / additional info to 'areeve @ reevesoft.com' and I'll try to incorporate them.

Computers

Acorn Archimedes (Archie)
Acorn Atom
Acorn Electron
Altair 8800
Amstrad CPC 664/6128
Amstrad PCW
Apple I
Apple IIe
Apple Macintosh Plus
Atari 800/XL/XE (8-bit)
Atari ST
Bandai RX-78
BBC Micro Model B / Master (128k)
Commodore 16 and Plus/4
Commodore 64
Commodore Amiga
Commodore PET
Commodore VIC-20
Interact Home Computer
Jupiter Ace
Mattel Aquarius
MSX/MSX2/MSX2+/MSX3
NEC PC-88
Orao
Oric-1 / Oric Atmos
PC 486DX 33 (AO486)
PDP-11
SAM Coupe
Sinclair QL
Sinclair ZX Spectrum
Sinclair ZX80
Sinclair ZX81
Spectravideo SV-328
TI-99/4a
TRS-80 Color Computer 2
TRS-80 Color Computer 3
TRS-80 MC-10
TRS-80 Model I
X68000
ZX Spectrum Next

Consoles

Adventure Vision (Entex)
Arcadia 2001 (Emmerson)
Astrocade (Bally)
Atari 2600
Atari 5200
Atari 7800
Atari Lynx
ColecoVision
Fairchild Channel F
Game Boy / Game Boy Color
Game Boy Advance
Intellivision (Mattel)
Interton VC 4000
Neo Geo Advanced Entertainment System (AES)
Nintendo Entertainment System (NES)
Odyssey II (Magnavox)
Sega CD / Mega-CD (Sega)
Sega Genesis / MegaDrive
Sega Master System (SMS) / Game Gear
Sony PlayStation
Super Nintendo (SNES)
TurboGrafx-16
Vectrex
WonderSwan / WonderSwan Color

Acorn Archimedes [Archie core] (1987)

The Acorn Archimedes is an ARM based system that contains its Operating System (OS) on ROM chips so just boot up and after the OS initializes you'll land in the Desktop. The github page for this core has the RISC OS 3 ROM so we'll assert that you are using that. This core supports an Archimedes computer with 4 megabytes of memory.

The Archimedes mouse had 3 buttons and, as PC mice usually have just two, you can use the OSD to make the right mouse button either button 2 or 3. The left mouse button on a PC mouse is always the select button (and I'll often just call this clicking). The right mouse button is the adjust button by default (swap buttons in the OSD is off), but can also be the menu buttons (swap buttons is on). The menu button seems a bit more useful in apps as clicking on the app with the menu button is often how one would access options such as save.

Keyboard Mapping

This computer's keyboard looks almost the same as a PC keyboard.

Win+F12 = OSD (as this computer has an F12 key)

RISC OS

Once the RISC OS loads you'll be on the Desktop and see an icon bar across the bottom of your screen with two icons for floppy drives on the left and icons for programs on the right. Next to the floppy icon will also be an icon for a few applications that are stored in ROM. You can click on that icon to open a window and see what's there. You can also insert a disk image in a floppy drive via the OSD and then click on the floppy drive icon to see a window with the files stored on that disk. You can then double-click on programs (which have names prefixed with an !) to load them. Once loaded, window based applications will add an icon to the icon bar on the lower right and you will be able to click on it to activate the program. Applications (many games) that don't use the GUI will just load and run.

The Desktop also has a command line interface that can be accessed via many ways giving access to what are called 'Star Commands' (because the Command Line prompt is a *). One way to enter Star Commands is by simply pressing the F12 key. You will see a * prompt at the bottom of the screen (which indicates the command line interface [CLI] is awaiting your command) and the entire screen will be used for commands. You can press the Return/Enter key on the * prompt to exit back to the Desktop. You

could also, instead, hover over the Acorn icon (the rightmost one) on the icon bar, click on the mouse's menu button, and choose Task Window which will open a window with a * prompt. Lastly, you could also choose New Task from the Acorn icon's menu which will let you type in a single star command.

More information on Star Commands can be found here:
http://www.riscos.com/support/developers/prm_index/starcomms.html

BASIC

BBC BASIC V is built into the ROM. To access it just type `BASIC` at a command line prompt. To exit BASIC type `QUIT`. BBC BASIC is VERY powerful giving you access to OS level function calls and inline assembler to optimize your BASIC programs. You can save you program with the SAVE command (e.g. `SAVE "filename"`) and, surprise, loading a program is done via the LOAD command (e.g. `LOAD "filename"`) however the structure of a filename needs further explanation.

A filename consists of the following format:

[FileSystemName:][DiskName].Filename

and for a floppy disk the file system name is "ADFS". The disk name is chosen when the disk is initialized and I found a blank image with a disk name of "Blank". The "." is used to separate folder names so the command

```
SAVE "ADFS:Blank.Test"
```

would save the current BASIC program in the root directory as the name "Test" on the floppy disk named "Blank". There is also a * command called DIR which allows for the setting of a default current path which should allow just typing `SAVE "Test"` (DIR, unlike on a PC, does not list the files on the disk.

More information on BBC BASIC can be found here:
http://www.riscos.com/support/developers/bbcbasic/

.ADF files (floppy disk images)

- Use OSD to insert a floppy disk image
- Click on Drive 0 (bottom left) and a window will open with the disk's contents
- Double-click on program icon to load

Note: The Archimedes' RISC OS will keep track of floppy disks so you can insert one disk image, double click to view its content in a window, insert another, double click to view its contents as well, and then double click on a program from the first window. You will be prompted to put the first disk back in the disk drive.

.HDF files (hard disk images)

This core does support hard drives, but you'll need a valid hard drive image and to follow the instructions on the github page for this core:

- Delete CMOS.dat from the Archie folder
- Reload the Archie core
- From a command line star prompt (press F12) type the command `configure idefsdiscs 1`
- Select the hard disk image (via the OSD)
- Do a cold reset

This will add an icon for the hard drive on the icon bar. I finally found a hard drive image that works with this core after several that did not so I can confirm it works, but I am unclear as to the exact format of this image file. I also tested some non-working images I found against a few software emulators which required the parameters of the hard drive (which this core does not) so I have to think this core uses a special format or something to determine the geometry of the hard drive. I don't know how to create a fresh, blank hard drive image to work with.

Acorn Atom (1980)

The Acorn Atom is a 6502 based computer that was available in kit form or as a complete unit with 2K (expandable to 12K) of RAM and a cassette tape interface for storage. The MiSTer Acorn Atom core provides 32 KB of RAM and in addition to (untested) cassette support includes the AtoMMC2 ROM which permits used of an SD card for mass storage (because it's on MiSTer this means a virtual SD file on an SD card).

Keyboard Mapping (using the UK layout, but other layouts are available via the OSD)

Lock = Caps Lock
Repeat = Right Alt Key
Copy = Tab
Break = F10
* (looks more like a diamond) = Shift+8
Up Arrow = Shift+6 (because the Shift key is already used for an Up Arrow you would need to use the Left Alt key version of Shift as well for a Shift+Up Arrow)

The Break key (F10 on a PC keyboard) is used to 'restart' the ATOM and does different things based on keys pressed along with it:

Break = Show the Atom ROM type
Shift+Break = Mount the VHD (and automatically execute a program named MENU)
Ctrl+Break = Quit / disable the MMC ROM
Repeat+Break = Show AtoMMC2 firmware version and type of storage card inserted

Storage

While this core may support simulation of the cassette tape interface it primarily relies on using .VHD files which use the FAT FORMAT and are mounted via the AtoMMC2 ROM. You can find a sample .VHD file in the MiSTer Acorn Atom github and you can mount that file on your PC and copy files to it because using it with the Atom core. However, chances are any files you might want to copy are already a part of the amazing Atom Software Archive on github (link below) that has most Atom Software all packaged together with a bow on top (aka a menu program) as a .VHD file. Just mount the image and press Shift+Break (Shift+F10) and it'll automatically boot into the menu (and press / in the menu for help).

BASIC

When first starting the Atom without any automatically loading software you will see the > prompt for BASIC. You can load and save a BASIC program with the usual load and save commands

```
LOAD "filename"
SAVE "filename"
```

But you can also execute DOS (Disk) commands via preceding them with an asterisk (Shift+8):

`*CAT <optional folder name>` = Show a disk directory (catalog) on the screen
`*INFO <filename>` = Show file details such as the load address, execution address and length
`*RUN <filename>` = Run a program

Games

I had mixed results with figuring out the controls to arcade games on this system. I tried a Galaxian clone from the Atom Software Archive and to control my ship I had to use the Lock (CapsLock) key to move left, Left Shift key to move right and Repeat (Right Alt key) to fire. I found several other keys to move right, but just those two for move left and fire. I also tried setting up a joystick from the OSD, but not luck (the Atom offered a keyboard based joystick and also had an add-on via the AtoMMC).

Links with more information

AtoMMC2 info at http://www.acornatom.nl/atom_plaatjes/sd-files/atommmc2.html

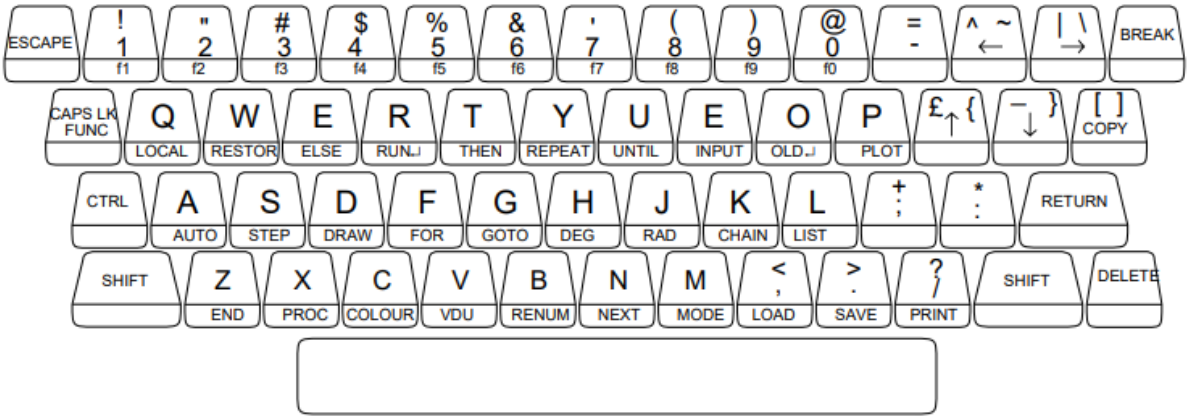
Atom Software Archive at https://github.com/hoglet67/AtomSoftwareArchive/releases

Atom Disc Pack Manual (other DOS commands)
http://www.acornatom.nl/sites/fpga/www.howell1964.freeserve.co.uk/acorn/atom/atom_fdc.htm

Atom Programming in BASIC and Assembler
http://www.vintagecomputer.net/fjkraan/comp/atom/atap/index.html

Acorn Electron (1983)

Nicknamed “the Elk”, the Acorn Electron was Acorn’s response to the low cost ZX Spectrum. Acorn had a foothold in the education market with their partnership with the BBC, but the BBC Micro was pricey so the low cost yet very similar Acorn Electron was released so that the masses could afford one. It spoke the same BASIC that school kids were learning, but the two are just different enough to be their own computers. The Electron was also intended to be quite expandable. The base model Electron came with 32K of RAM and a full size keyboard.



The above keyboard picture (from the Electron User Guide) serves to show the layout of the 5 special keys (4 arrow keys and the copy button) along with the commands/function keys that were printed on the front of many keys. The PC’s Caps Lock key acts as the Function key on the Electron’s keyboard so pressing Caps Lock+K would be equivalent to typing in the CHAIN command printed on the front of the K key. Furthermore, the 10 function keys (F1-F10) are typed via Caps Lock+1 through Caps Lock+0. The [and] are used for the up/down arrow keys, but you can also use the actual arrow keys on the PC keyboard for those and must for the left and right arrows.

Before we jump into using the Electron let's note that this core supports an MMFS device which is a flash memory device which allows SD cards to act like floppy disks. When you first boot the Electron core (assuming you have not added a boot.vhd file) it will prompt you for an SD Card with a 'Card?' prompt and it will seem like you can't get very far with the keyboard with all keystrokes being ignore. It's looking for an SD Card in the MMFS device. From here you can either use the OSD to mount a .VHD file that acts like a memory card (discussed later) or you can pretty Control+F10 on your PC keyboard to boot to a regular prompt in BASIC (you could start typing in a BASIC program).

Loading Software from Cassette Tape

The Electron core supports the ADC to load software from real cassettes and it supports .UEF files which are Electron cassette images. In order to load a program from cassette

- Load the .UEF file via the OSD and return to the Electron
- Type `*TAPE` to make using the cassette tape drive the primary device
- Type `CHAIN ""` to load and run a BASIC program

Note that most Electron programs would have a small BASIC loaded program and that would then run, display some info on screen and then initiate loading the machine language program. The core also displays a cassette overlay on screen to show you that the core's simulated cassette drive is at work.

Loading Software from Floppy Disks/VHD files

Floppy disks were expensive and most Electron software came on cassette tapes, but we're in the 21st century so naturally there's a memory card based solution (the ElkSD-Plus) for the Electron and this core has that built in. The way this works is that a file (often named BEEB.MMB) contains the contents of hundreds of floppy disks and you can select which of the simulated floppy disks are in use on your Electron at any given time via 'star commands' (commands that being with a star or asterisk). You can build your own BEEB.MMB file from the many .SSD floppy disk image files available with a utility named Beeb Image. Why do we care about BEEB.MMB fil? Because they are the same .VHD files that this core uses so you can simply rename your BEEB.MMB file to GAMES.VHD and use it here (or you could name it BOOT.VHD and then it will automatically load with this core).

Once you have your VHD file with multiple images in it the following star commands should help:

`*MMFS`	Make the SD Card device the primary device (for star commands like *CAT to see what is on them)
`*TAPE`	Make the tape device the primary device
`*DCAT <optional search filter>`	List all virtual floppy disks stored on this SD Card
`*DDRIVE`	List all virtual floppy disks inserted into the Electron's 4 disk drives
`*DIN #/name`	Insert virtual floppy disks into one of the Electron's 4 disk drives
`*DBOOT <disk number or name>`	Insert the specified virtual disk into drive 0 and auto-boot it (if the disk can be auto-booted)

A more complete MMFS command reference can be found here:
https://github.com/hoglet67/MMFS/wiki/Command-Reference

BASIC Programming

As stated in the introduction, BASIC on the Electron is basically (hah!) the same as on the BBC Micro and it's quite good. While you can't load/save to tape (maybe you can with the ADC?) you can load/save on the virtual floppy disks provided by the MMFS support. Furthermore, this version of BASIC allows you to program in Assembly/Machine language easily within your BASIC programs. It's pretty impressive. While I won't attempt to teach BASIC programming here I will provide commands for loading and saving and a link to a reference that contains extensive info on programming the Electron.

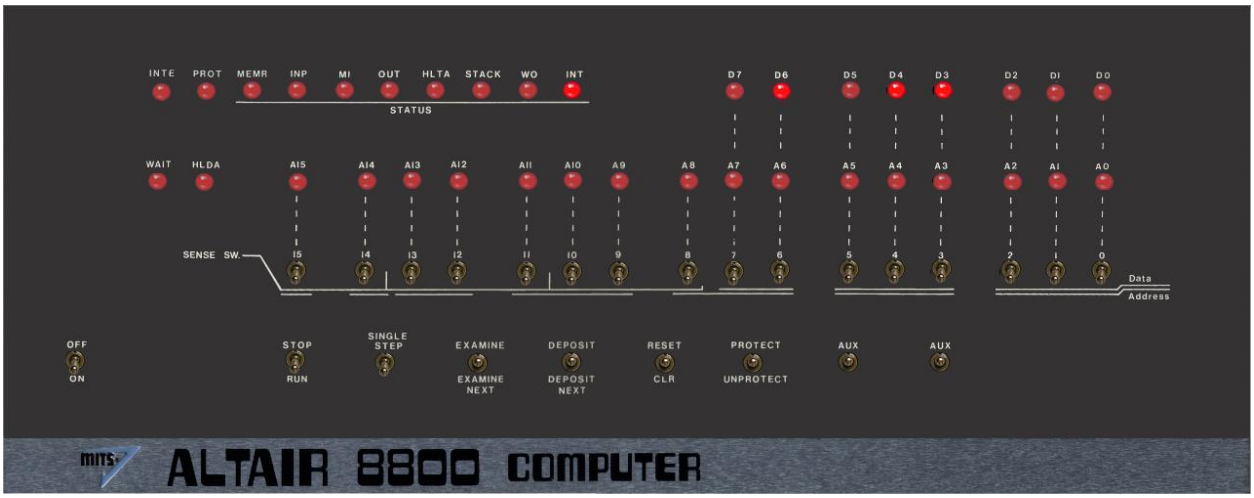
`LOAD "name"` = Load a BASIC program from the primary device with the filename name.
`SAVE "name"` = Save a BASIC program to the primary device with the filename name.
`CHAIN "name"` = Load and run a BASIC program from the primary device with the filename "name".
`*SAVE "name" <start addr> <end addr> <exec addr>` = Save the machine code from the start through the end address with initial execution address as provided.
`*RUN "name"` = Run a machine language program saved via the *SAVE command.

Learn more about using the Electron and programming it in BASIC (and Assembly Language) here:
http://www.adventuresinretro.co.uk/wp-content/uploads/2014/04/Acorn-Electron-User-Guide.pdf

Gaming

There are a lot of games for the Electron and, as the joystick was not standard on the system, many require using the keyboard for control. I didn't notice any real standard for specific keys correlating to left, right, up and down so that seems to be on a per game basis.

Altair 8800 (1975)



Use cursor to select switches and press 1 key (up) or 0/2 key (down)

In order to run one of the included programs (external programs are not supported):

- Select the program
- Navigate to On/Off and press 2 to flip the switch that turns the Altair 8800 on
- Ensure that Stop/Run switch is in Stop Mode
- Use OSD to select and load the program
- Navigate to Reset/CLR switch and press Reset (1)
- Navigate to Stop/Run and press Run (2)

and watch the LEDs flash.

Amstrad CPC 664/6128 (1984)

Commands (not case sensitive)

Note: The pipe symbol or '|' is used in many commands and is typed via Shift+@

`CAT` or `|DIR` = List disk files
`RUN"filename` = Execute a program named filename from the disk
(often `RUN"DISC.BIN` or `RUN"DISC.BAS` where including BIN or BAS are optional)
`|A` or `|B` = Switch to drive A or drive B

Special MiSTer specific core key combos:

Alt+F1 = Toggle mute on cassette sound
Alt+F2 = Unload the tape

Loading Cassette (.CDT) files?

- Type `|TAPE` to enter cassette mode when disc interface is present
- Type `RUN"` (or press Shift-Enter)
- Press any key as prompted

Loading a disk with no directory/catalog?

Type `|CPM` to load disks with no directory

Loading expansions?

Amstrad PCW (1985)

The Amstrad PCW was initially conceived as a low cost word processor, but the games eventually showed up with the system's success. It offered a hi-resolution monochrome display and you can choose your foreground color in this core's settings (from the images I see a green screen was typical).

This machine ran CP/M and came standard with a disk drive (this core supports .DSK images). Often, but not always, .DSK image games come in pairs with a boot disk and a game disk. After booting the boot disk (select disk image and choose Reset both from the OSD) you'll be asked to swap in the second disk (via the OSD) and to press Enter once done.

Apple I (1976)

The Apple I is more of a piece of history than anything as it only lasted for about a year until it was discontinued because the far more popular Apple II had been released. However, not too many computers end up in the Smithsonian and now you can tinker with your own Apple I.

The Apple I MiSTer core had the 6502 Monitor (WozMon) and BASIC preloaded into memory. It boots to a \ on a line followed by a blinking @ (which is the cursor) on the line below. You're now in the 6502 Monitor and ready to start peeking around at memory and even entering machine code programs.

Type

```
0 : A9 0 AA 20 EF FF E8 8A 4C 2 0
```

on a line followed by the Enter key and that will insert the following 6502 machine code program into the Apple I's memory (I have listed the hex op codes entered as comments after a ; on each line)

```
LDA #0      ; A9 0
TAX         ; AA
JSR $FFEF   ; 20 EF FF (OS routine to print the char in the A register)
INX         ; E8
```

```
TXA         ; 8A
JMP $0002   ; 4C 2 0
```

You can now inspect memory to verify that what was typed is in memory by typing

```
0 . A
```

which is the command to dump memory from a start address ($0000) to an end address ($000A) (with $ used to indicate hexadecimal numbers are being used). You could try another memory dump command too. Just enter any two memory addresses in hexadecimal with a dot in between (the spaces are optional).

Now comes the fun, let's run the program. Just type

```
0000 R
```

Where 0000 is the hexadecimal address to start running your program at and R means to run it (you could also type R by itself which means run at the 'current' address, but this is the more generic command). This program just prints all the characters in the character set to the screen over and over and over. You'll have to Reset the computer to get out of this which is done via the OSD or the User Defined Button on the MiSTer. At this point the screen might be a mess. There's supposed to be a Clear Screen button, but I couldn't figure that one out… maybe it's just not part of this core. You can just hit Enter several times to scroll stuff off of the screen. Next let's use BASIC. Type

```
E000 R
```

because the BASIC interpreter resides at addresses $E000-$EFFF. Each line will now have a > before the flashing @ cursor. You can type in BASIC programs and run them.

Loading programs is done via the OSD however this core uses plain text files. In a nutshell loading program is essentially as if whatever is in the text file is being typed in via the keyboard. You can stick machine language programs or BASIC programs in a text file and load them provided that you're in the right content in the system (e.g. you need to be in BASIC before loading a BASIC program… it's literally going to be as if you've typed in whatever is in the text file).

Reset is then used to exit from BASIC back to the 6502 Monitor. If you had a BASIC program in memory the Reset button does not erase it… you do need to run BASIC by running at address E2B3 (instead of E000) though.

Apple I User Guide
https://mirrors.apple2.org.za/ftp.apple.asimov.net/documentation/apple1/apple1manual_alt.pdf

Apple I BASIC Users Manual
https://archive.org/details/apple1_basic_manual

Apple IIe (1977)

The MiSTer's Apple IIe core has support for one floppy drive (read only), one hard drive (read and write) and a Mockingboard sound card (among other features). Also note that the Apple IIe's Reset button (upper right of the keyboard) is supported via the I/O board's 3rd (core specific) button vs a keyboard key.

Many Apple II disks are auto-boot and just require insertion via the OSD and a reset to load. However there are some games such as the Eamon series that are written in Applesoft BASIC and need further instructions:

- You'll need to boot from an Apple DOS 3.3 disk which will load DOS.
- Once DOS loads you'll receive the familiar `]` prompt that Applesoft BASIC uses.
- Insert the game disk via the OSD.
- Type `CATALOG` to see the files on it. The letters to the left of the file name such as A and B mean the following: (A=BASIC program, B=Binary (machine language) program, T=Text file).
- For BASIC programs load via typing `LOAD <name of program>` or `RUN <name of program>` (to load and automatically run the program). If you load it then you'll be able to type LIST and see the BASIC program code and such before you type RUN to execute the program.
- For Binary programs type `BRUN <name of program>` to execute them.

DOS

The Apple II line of computers originally used Apple DOS which has versions 3.1, 3.2 and 3.3 (unless you have good reason stick with 3.3) and ProDOS (later called ProDOS 8 for the 8-bit Apple II [and to differentiate it from ProDOS 16 for the Apple IIgs]) which added several new features such as the concept of subdirectories, but, more importantly supported hard drives. DOS would typically load from a boot disk and you would be able to use some additional DOS commands from the BASIC `]` prompt to do things like load and save programs and get a list of files on the disk. More advanced tasks such as copying files required separate utilities.

BASIC commands

`PR#6` = Boots the disk in floppy drive 1 (slot 6 on the Apple II which is where the 6 comes from) [Note that I see a £ symbol instead of # in this core telling me it's a British Apple II, but it appears to serve the

same purpose as the # sign)
`CALL -151` = Enter the machine language monitor

The Machine language monitor displays address/register info ala
`9BD6 - A=01 X=BA Y=54` … etc

Followed by an `*` prompt (which tells you that you're in the monitor and not Applesoft BASIC) instead of a `]` prompt.

Type `E000G` which means go to address E000 (and return to Applesoft BASIC [with or without DOS loaded])
Ctrl-C followed by <Return> exits back to BASIC leaving any BASIC programs intact
Ctrl-B followed by <Return> exits back to BASIC but clears program memory

Apple DOS 3.3 Commands

`LOAD <name of program>` = Load BASIC program
`RUN <name of program>` = Load and Run BASIC program
`SAVE <name of program>` = Save BASIC program (remember, writing to a floppy disk is not yet supported)
`BRUN <name of program>` = Runs a binary/machine language program from disk.
`CATALOG` = List files on the disk with entries that look like

```
*A 003 HELLO
```

where the first char is a space (not locked) or an asterisk (if the file is locked [and cannot be accidentally deleted]), A = BASIC/B=Binary/T=Text file, 003 = the file uses 3 256 byte blocks of storage on the disk, and HELLO is the name of the file.

ProDOS

If you want to use a hard drive and be able to save data then you'll need to use ProDOS, but first you'll need to create a virtual hard drive file (.hdv) and install ProDOS onto it so you can boot from it. Adding a hard drive to your MiSTer FPGA Apple IIe setup requires a few steps.

1) You'll want CiderPress which is a utility to create Apple II disk images (http://a2ciderpress.com/)
    a. Once you have CiderPress installed use the 'File… New' option and create a 32MB ProDOS disk image.
    b. That will create a .PO (ProDOS order) disk image, but you'll need to convert that to a .HDV disk image for MiSTer so choose 'Tools… Bulk disk image converter' and select the .po file you just created. Next choose where you want to save the converted .hdv file and finally choose the "Sim IIe" virtual hard drive option to make a .hdv file from the .po file.
    c. You're ready to go with a fresh Apple IIe hard drive!
2) Next you'll want to install ProDOS onto this hard drive image so you can boot from it and to do that you'll need ProDOS. I used this one (ftp://ftp.apple.asimov.net/pub/apple_II/images/masters/prodos/PRODOS-8%20v4.0.2%20System.dsk) as per these instructions: http://www.vintage.excited-geek.com/blog/recipes/apple-ii/emulation/creating-a-bootable-prodos-hard-drive-image/.

3) Mount this hard drive image from the OSD.
4) Mount and boot the ProDOS floppy disk image, but in order to do that you'll need to use the MiSTer OSD and switch to the 65C02 CPU otherwise it'll tell you you'll Apple IIe isn't the enhanced Apple IIe that is required to run ProDOS.
5) Load the System Utilities.
6) Format the hard disk. The hard disk will be in slot 7, disk 1 (the floppy is slot 6, disk 1).
7) Copy the PRODOS and BASIC.SYSTEM files from the ProDOS floppy disk image (slot 6, disk 1) to the virtual hard disk image (slot 7, disk1).
8) and now you have a bootable hard drive with ProDOS installed on it.

From here you can boot from the hard disk image and both load AND SAVE programs. Go ahead, type in a simple BASIC program and then type `SAVE TEST` and it will save to the hard drive. Type `CATALOG` (or `CAT` in ProDOS as a shortcut) to see file on the disk and you'll see your TEST file listed. Next type NEW (to clear your program from memory), type `LIST` to convince yourself that it's gone, and then type `LOAD TEST` to load it back from the hard drive.

A few last notes…

If you are using ProDOS with both a hard disk image and a floppy disk image you can get a CATALOG of the 'other' drive by adding ',S#' to the end of the command where # is the slot of the disk controller (6=floppy, 7=hard drive) and when you do that it will be sticky so future disk operations will be directed do that device until you change it with this add on to a disk command e.g.

CATALOG,S6

will give you a directory of the content from the floppy drive. There's also a ',D#' command for when these cards have two disk devices attached to them, but as of right now I only see that it's possible to have one floppy and one hard drive via MiSTer.

And no, it's not easy to copy files from an Apple DOS 3.3 disk to a ProDOS disk. The two are incompatible so you'll need an Apple II utility program such as Copy II+ 7.2 to assist with that.

Apple Macintosh Plus (1984)

The Macintosh Plus was the third release (1986) in the Macintosh line of computers and came with a 68000 CPU and 1 megabyte of memory. The Macintosh floppy disk drives were different from most other computers as one could not just eject a floppy disk with a button as on a PC, but instead had to choose to eject the disk from the GUI interface. Mac OS would keep track of disks that had been in the drive so instead of icons correlating to floppy disk drives the icons would correlate to disks that had been recently inserted/ejected and, when accessed again, the computer ask the user to reinsert said disk. Also when inserting a disk it may take several seconds for the Mac to mount the disk and for its icon to initially appear. Floppy drives on the Macintosh also used a different low level format making them incompatible with PCs at a hardware level. The original floppy drive on the Macintosh could store 400KB of data, but the Mac Plus added double-sided floppy drives to increase that capacity to 800KB.

Note: The Macintosh line of computers has a lengthy history starting off with a simple 68000 based black and white computer sporting among the first graphical user interfaces (GUIs) geared at the mainstream consumer. However later models added color support and we've seen Power PC, Intel and Apple M1 ARM CPUs. This core is solely for a mid to late 1980s Mac Plus.

Once your disk has mounted in the GUI and its disk icon appears it should be easy to open the disk and double-click to run a program.

Getting Started

If you start the Mac Plus core you'll be greeted with an icon of a floppy disk with an X through it indicating it cannot locate any System software. You can get a System 6.0.5 Startup floppy disk image directly from the core's github page along with an empty hard drive image. If you boot from the floppy with the hard drive image installed the Mac will see it after loading the System software from the floppy drive and ask if you want to initialize the hard drive. Initializing the hard drive will allow you to run the Installer from the System Disk so that you can install it to and boot from the hard drive.

Keyboard Mapping

Alt key = Macintosh Command key (to the left of the space bar in the picture)
Windows key = Macintosh Option key

.DSK files

This core uses .DSK files which are just raw dumps of the sector data of a floppy disk much like the .IMG format on the PC core. These can be inserted via the OSD, but remember to Eject the disk via the Macintosh Desktop before inserting another disk.

.VHD files

This core uses .VHD files for hard disk support and the github repository for this core includes an empty 20 MB hard disk image.

Atari 800/XL/XE [aka Atari 8-bit] (1979)

The Atari computers started with 8K/16K of memory, but 48K quickly became a standard. With the release of the XL line of computers Atari allowed the 10K that the OS originally occupied and the unused 4K of address space in the 400/800 to be banked out and replaced with RAM giving access to almost a full 64K (2K was used for hardware registers). Later attempts to add even more memory meant bank switching from 3rd parties which translated to a lack of a standard. This core also supports several of the more popular methods for bank switching, but for most software 64K should do.

Keyboard Mapping

On the right side of the Atari XL keyboard (above) are 5 keys: Help, Start, Select Option and Reset. Help appears with the XL line of computers and was never really used outside of the built in diagnostics on those computers. These keys can be accessed via a PC keyboard as follows:

F5 = Help
F6 = Start
F7 = Select
F8 = Option (hold down on boot when in XL/XE mode to disable BASIC ROM)
F9 = Reset (Warm Start)

F10 = Cold Start (aka power cycle)
F11 = Select and disk image for drive 1 and reboot (load selected disk)
Shift+Control+N/H = Disable/Enable high speed SIO (aka speed up disk I/O)

ROM OSs

In the OSD you can select between OS A, OS B, XL OS and the XL OS without BASIC. OS A was an early OS that was quickly replaced by OS B for the Atari 400/800. When the XL line of computer came out this OS was replaced with the XL OS. The XL computers also had a built in BASIC ROM (see What's the deal with BASIC) which could be disabled by holding down the Option key upon booting OR you can choose XL OS without BASIC to disable it.

Note: In the OSD one can select between OS A, OS B and the XL OS. Usually you'll want the XL OS, but there may be some older games that used undocumented OS calls that require OS B (or OS A although that's quite rare). There was a translator disk for XL/XE computers to address this issue, but no need for that with MiSTer.

Startup

When initially running this core you'll end up in one of three places. At a BASIC 'READY' prompt if you have the BASIC ROM inserted, in the Self Diagnostic tool (on an XL/XE without BASIC) or in MEMO PAD mode (on a 400/800 [use the OS B instead of an XL OS ROM]). The initial gurgle sound is the computer looking for a disk to boot from and failing to find one. You could also hold down the Start key at bootup to try and load from cassette, but cassettes are not supported in this core.

DOS

There are several Disk Operating Systems for the Atari 8-bit computers, but Atari's DOS 2.5 was probably the most popular. It was a menu driven DOS and unlike several other computers from this era you would not run machine language programs from Atari BASIC. Instead you would need to type DOS from Atari BASIC's READY prompt to return to the DOS menu (which would take time to load from the disk) and then you would choose the Run Cartridge option from DOS to return to BASIC. From BASIC you would benefit from DOS providing file system support and could, of course, load and save BASIC programs via the LOAD/SAVE commands specifying which disk drive to load from/save to e.g. `LOAD "D1:MYFILE.BAS"` and `SAVE "D1:MYFILE.BAS"` where D1 meant disk drive #1 (the only one in a single drive system). You could, of course, run DOS without BASIC (or any cartridge) and in many cases (where machine language programs would occupy the memory use by ROM cartridges) would have to.

Load .ATR/.ATX (supports .ZIP files):

- Insert bootable disk image via OSD (.ATR files are sector based disk images, .ATX files are a sophisticated format meant to include copy protection to yield the most authentic experience possible).
- Press and hold F8 (the Option key to disable BASIC on XL OS unless BASIC is required) while pressing F10 (Cold start) to simulate turning the computer on with the Option button being held down (optionally you can also choose XL without BASIC from the OSD, but as an old Atari owner holding down Option feels more authentic ☺.
- Note: If the disk image isn't bootable (most should be as there were many tiny menu programs available) then you might need to load DOS and run the program from there. .EXE was the most common file extension for machine language programs, but .COM would also be used.

.XEX files

.XEX files are renamed Atari 8-bit .EXE machine language program files, but to disambiguate between a PC .EXE file and an Atari 8-bit .EXE file they were renamed (get it, the opposite of EXE is XEX… kind of). This core will load and run them just like an Atari DOS would by selected them from the OSD.

Load .CAR/.ROM/.BIN ROM cartridge image

Select a cartridge image via the OSD and the core will automatically reboot. .ROM and .BIN files are just raw dumps of the cartridge data, but .CAR files have a header to provide additional info about the cartridge. The Atari 8-bit computers natively support 16K cartridges so anything bigger than that will require some sort of bank switching method. Sometimes this core will not support a particular cartridge format and it will tell you. Other times it won't immediately know what to do and will prompt for more info as to how to use the cartridge.

What's the deal with BASIC?

On the original Atari 400/800 the BASIC programming language was a separate cartridge so if you needed BASIC you inserted the cartridge. The idea behind this was that perhaps you'd rather program in PILOT or Assembler so there are cartridges with those languages on as well. The Atari XL series changed that and built the standard BASIC cartridge into the computer's ROM however BASIC occupies some memory and if another program needs that memory then you'd be out of luck so the decision to insert/remove the built-in BASIC cartridge remained and that selection is made by holding down the Option key while powering on the computer.

Atari ST (1985)

This is quite a full featured Atari ST core supporting many different configurations of the Atari ST computer. It supports two floppy drives and ACSI (Atari Computer Systems Interface) [Note: Similar, but not compatible with SCSI] hard drives as well.

Ensure there's a version of TOS selected and the core will boot into the Atari ST's GEM Desktop user interface that sports the usual icons for floppy drives (which correspond to physical hardware and not disks being present in them), drop down menus and windows showing the contents of a disk upon double-clicking on the floppy drive icons.

Many floppy disks will auto boot because they have a folder named AUTO and it contains programs that will automatically run upon booting from said disk so you can either insert the virtual floppy disk and reboot or insert the disk after booting, double click on the appropriate floppy disk icon and run whatever is in the AUTO folder. Files with an extension of .PRG and .TOS are executables that can be double-clicked on to run them.

The Atari ST had two 9-pin 'joystick' ports, but the first was usually occupied by the mouse so only the second was available for a standard DE-9 Atari joystick.

The Atari ST supported 3 graphics modes (low, medium and high) and Atari made two CRT monitors available: one for low/medium (color) mode and one for high (monochrome) mode. If you're just interested in playing games then it's pretty safe to stick with the low resolution, color mode. You can change between using color (low and medium) or monochrome (high) resolution modes via the MiSTer OSD under Options->Screen. If you're in color more on the Atari ST Desktop you can change between low/medium resolution from the GEM Options menu.

.ST files

.ST is the extension used for Atari ST disk images. You can insert them via the OSD and then try to autoboot them by choosing Reset. Depending on the disks's content it may boot or it may land you back at the GEM Desktop possible with some Desk Accessories (little extra programs with an .ACC file extension available under the Desk menu).

BASIC

The early (not certain about later) models of Atari STs came with an Atari ST Languages Disk that included Atari ST BASIC. BASIC is just another program to run and is called BASIC.PRG on the ST Languages disk. Just insert the virtual disk image and double-click on the BASIC.PRG (after double-clicking on the floppy drive icon to see the BASIC.PRG file on the disk). I was able to type in a short BASIC program, run it, save it to the disk and later reload it (and unlike on many cores I was also able to take an existing disk image and reformat it to have a fresh, blank disk). Note that ST BASIC isn't very good and most BASIC programmers probably switched to something like GFA BASIC.

.VHD files

.VHD files are virtual hard disk images. You can create these via Windows 10's Disk Management Utility, but you'll probably need to copy some drivers onto them. Furthermore, the GEM Desktop does not seem to automatically add icons for hard drive partitions even once it recognizes their presence via a drive. You will need to add that by clicking on a drive icon and selecting the menu item to add a drive icon from the Options menu. The partitions are there so just enter the appropriate drive letter and the text label you want to use. Once the icon has been added you can choose the Save Desktop menu item and then these icons will be there the next time.

Atari ST Models

These settings should simulate the following Atari ST models that were released. Note that the only way to tell the difference between TOS version inside the Atari ST's GEM Desktop is by looking at copyright dates in the About dialog box. It did not display version numbers.

520ST/1040ST: 512KB/1MB RAM, TOS 1.0, no BLiTTER

STf and STfm models: These just added a built in floppy drive and RF modulator. There's nothing to see here with respect to MiSTer support.

Mega ST 1/2/4: 1/2/4 MB RAM, TOS 1.02 (added BLiTTER support)/1.04, added the BLiTTER chip

520STe/1040STe: Same as 520ST/1040ST but added the BLiTTER chip and the STe chipset, TOS 1.06/1.62

Mega STe: Same as Mega ST models with the MegaSTe chipset, TOS 2.02-2.06 Note: This model had a 68020 CPU that could be switched between 8 MHz (more compatible) and 16 MHz clock speeds which I do not see here.

Core Options

The BLiTTER chip was a chip that did high speed memory transfers often for graphics oriented data.

This core also supports the Viking chip, but I am unsure of what that is.

Bandai RX-78 (1983)

The Bandai RX-78 is a Japanese, Z80 based gaming computer from Bandai. Games were released on cartridge and cassette (not currently supported by the MiSTer core) media, but the software library is relatively small with only about 20 commercially released game titles along with a handful of educational and application programs.

BBC Micro Model B / Master [128k] (1981)

Keyboard Mapping

Ctrl+F11 = Break key
Shift+Ctrl+F11 = Reset (with autostart if autostart is disabled)

Software is loaded via MMB files which are essentially VHDs (virtual hard drive images). Stick a .VHD image renamed to BEEB.MMB on your SD card and it'll boot. You can create your own MMB files, but it's not hard to find precreated ones that are quite comprehensive.

From BASIC there are commands to access the disk called star commands as they are all preceded with a star (asterisk) [see Acorn_DiscSystemUGI2.pdf (computinghistory.org.uk) for a complete list of commands]:

*. or *`CAT` = Show a disk directory
*`DRIVE X` = Switch default drive to Drive X (X = 0-3)
*`DIR <directory name>` = Change the current directory
`*RUN <file name>` = Run a machine language program from the disk

Only the SD cards in the secondary slot support writing
Some games (Uridium) only work on VGA (not HDMI) (bug?)

How to boot to BASIC (to enter the above commands)?

Commodore 16 and Plus/4 (1984)

The Commodore 16 core will boot to BASIC. It did have two joystick ports, but they were not the standard DE-9 (Atari style) ports however they were pin compatible so there are adapters (i.e. the joysticks were of the one button variety).

.PRG files (.PRG files are loaded directly into memory)

- Load .PRG file via the OSD
- Type RUN from BASIC

.TAP files (cassette tapes)

- From BASIC type `LOAD`
- Select .TAP file from the OSD
- Initially the .TAP will be searched for a program, when found there might be a small delay, but it will resume loading shortly thereafter
- You may need to type RUN once loading completes

.D64 (disk image files)

- Select the .D64 file from the OSD
- Type `LOAD "*",8,1` (`Shift+2 = "` and `] = *` on a PC keyboard)
- You may need to type RUN once the program has loaded

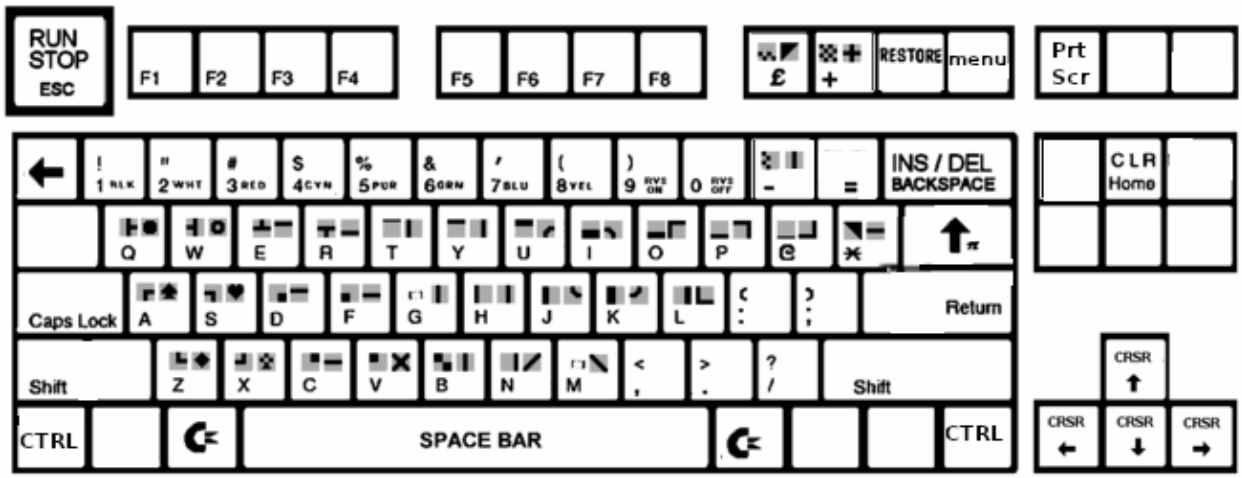
.BIN carts on Plus/4 (untested)

- Load .BIN file via OSD
- Press F2 from BASIC

Commodore 64 (1982)

Keyboard Mapping

F2/F4/F6/F8/Left/Up = Automatically activates the Shift key (as you'd need to do on a real C64)
F9 = Pound Key
F10 = + key
F11 = RESTORE key
Alt = Commodore Key



The Commodore 64 boots up directly into Commodore 64 BASIC displaying a familiar READY prompt allowing you to start typing in a BASIC program or load files from disk/tape right away.

BASIC

After typing in a simple BASIC program you can save it to a blank disk (unfortunately you can't prepare a blank disk in MiSTer… you'll need to add some blank disk images to your SD card beforehand) via the SAVE command e.g. `SAVE "FILENAME",8` where 8 is the device number 8 which is the first disk drive and you can later reload it via typing `LOAD "FILENAME",8`.

If you want to delve in further then I'd recommend taking a look at the excellent Commodore 64 Programmer's Reference Guide:

https://archive.org/details/c64-programmer-ref

.D64 files

Insert a virtual disk (.D64 file) via the OSD and from the BASIC READY prompt type the command `LOAD "$",8` (Shift+2 is the quotation mark on a C64 keyboard) to load a directory of files on the disk from device 8 (the first disk drive) followed by the command `LIST` to see the directory of files on the disk. Loading a machine language game (PRG file) from disk is done via typing `LOAD "NAME",8,1` (or leave the `,1` off for a BASIC program) followed by typing `RUN`.

.CRT files

.CRT files are ROM cartridges. Just select them from the OSD and the system will automatically reboot to run the cartridge based software.

.TAP files

.TAP files are cassette tape images. From the BASIC READY prompt just type `LOAD` and you will be prompted to press play on the tape drive. From the OSD select the .TAP file you want to load. You may also need to type `RUN` after it loads.

Commodore Amiga [MiniMig core] (1985)

The Amiga has 3 types of RAM: Chip, Fast and Slow. Chip is the main type of memory and is on the shared data. Fast RAM is only accessible by the CPU so it doesn't have to share the bus with any other chips making it faster with respect to the CPU. It's address space is also higher in the 68ks 24 bit address space (above address $200000). Slow RAM is located between chip RAM and fast RAM in the address space, sits on the shared bus, but is only accessible to the CPU.

Kickstart is the boot loader usually in firmware/ROM of an Amiga. The Amiga 1000 was initially released with version 1.0 on floppy disk, but later it was moved to ROM. The Amiga 500 was released with version 1.2 which would not boot from a hard drive. V1.3 added hard drive auto boot. Several later versions also exist.

Workbench is the Amiga's graphical file manager so basically it's the Amiga's version of what Windows/O S X would call a desktop. Initially it was released on floppy disk, but later versions could be installed on a hard drive. Together with Kickstart the Amiga's Operating System is former. Usually the Kickstart and Workbench were released in pairs with matching version numbers.

Settings for more common Amiga models:

Amiga 1000 (first model): CPU=68000, Chipset=OCS (Original chipset), Chip RAM=256/512KB, Kickstart/Workbench 1.0-1.3

Amiga 500: CPU=68000, Chipset=OCS/ECS (Enhanced chipset), Chip RAM=512KB, Kickstart/Workbench 1.2-1.3

Amiga 1200: CPU=68020, Chipset=AGA (Advanced Graphics Architecture), Chip RAM=2MB, Workbench/Kickstart 3.0

Note: The MiSTer github for the MiniMig core has some other suggested settings based on ECS/AGA modes… I am trying to get as close to original baseline Amiga models as I can with the above settings. You can always add more memory or try to use a later version of Kickstart/Workbench. There's no shortage of options here.

Workbench 1.3 comes on 2 disks. Try choosing the settings for an Amiga 500, select Kickstart 1.3 as your ROM image under Memory settings and insert disk 1 of Workbench into floppy drive 1 and disk 2 into floppy drive 2. Next choose Reset from the OSD to boot up your Amiga. It will initially boot to an AmigaDOS Window and eventually load a somewhat familiar GUI interface (Workbench) with familiar

disk icons for disk drives on the right side of the screen. From here you can click on the Workbench Extras disk and a window will appear showing programs contained on that disk. Most functions such as resizing, moving and scrolling the window's contents should be familiar, but accessing drop down menus is different. Drop down menus on the Amiga are accessing by right clicking on the window's title/drag bar. Also the two icons in the upper left are for moving the window to the background and foreground. From here load AmigaBASIC. You can also insert other disks via the OSD to browse their contents.

AmigaBASIC

AmigaBASIC on the 2nd Workbench 1.3 disk includes several sample programs that you could click on and run, but if you just run AmigaBASIC you'll see a BASIC window and a List window. The BASIC window is where you would type commands to run, list (view the source code) and save you program (although all of these tasks can also be done via the drop down menus access by right clicking on the title bar) plus it's where your program will output its results. The List window is where you BASIC program would be typed. Note that AmigaBASIC does not use line numbers. Try typing a simple BASIC program and selecting Run from the drop down menus.

.ADF files

Most Amiga floppy disk images use a .ADF file extension and auto boot meaning just stick them in the floppy drive, turn the computer on and they load. I tested on 3 games and 2 worked. One had two disks and I was expecting to be prompted to swap them. However, the game provided no help just quitting on a black screen with a mouse cursor, but worked once I put disk 2 into drive 2 via the OSD (in addition to disk 1 in drive 1). Because the Amiga has so many different configurations and settings you may have varying success and may need to fiddle with different settings to get various software to run. Note that the MiSTer has an LED that blinks indicating floppy disk access which helped me tell if a game load was in progress or, after lengthy periods of inactivity, was a fail.

.HDF files

.HDF files are virtual hard drive images for the Amiga. I don't see a way to create one with MiSTer so you would need an external tool such as the one UAE includes to create one of these however this core does support using them once created. You should also be aware of a program called WHDLoad which allows many floppy games to be installed on a hard drive. I see YouTube video on how to set this up and I see it used in pre-created hard drive images I have experimented with.

In the OSD you can adjust the current Amiga graphic mode's screen position with the following keypresses. This is done by selecting 'Adjust screen position' in the 'Audio & Video' settings. Once you are satisfied you need to return to this same location in the OSD and choose the finish option.

- Cursor keys = Top/Left
- Alt+Cursor keys = Bottom/right
- Enter = Finish and store position
- Backspace = Reset to default
- Esc = Cancel

F11 = Start monitor if HRTmon is enabled in the OSD (I have not seen this do anything when I tried using it so ?)

Commodore PET (1977)

.PRG files - These are injected directly into memory. Select/load via OSD and type RUN from BASIC.
.TAP files - Select from the OSD and press F1 to load.

Commodore VIC-20 (1980)

The VIC-20 is a bit of a pain due to it not being as simple as ensuring maximum memory in order to load any program. As memory of various sizes was added it would shift things around in the VIC-20's memory map (specifically the display would shift from the top of the default 5K available when adding anything more than a 3K memory expansion). This translates to sometimes having to get just the right memory configuration to load a given piece of software. It would be great if there was a chart containing this info, but I have yet to find one.

In order to get this core to work you will need a very specific boot.rom file that contains the C-1541 ROM, VIC-20 PAL ROM and VIC-20 NTSC ROM appended together into a single 32KB file. With that the VIC-20 would, like most computers from the early 80s, boot into BASIC.

Left Ctrl+Left Alt+Right Alt will reboot the VIC-20.

.PRG files (program files that will just be directly injected into memory)

- Select/load via OSD
- Type `RUN` from BASIC to execute

.TAP files (cassette tapes)

- From BASIC type `LOAD`
- Select the .TAP file to load from the OSD

.DSK files (disk images)

- Select .DSK file from the OSD
- From BASIC type `LOAD "*",8,1`

.CRT/CTx files (cartridges)

.CRT files have a header indicating where in memory to load them. CTx files do not and replace the letter x in the filename with a hexadecimal 2-B indicating at what memory address to load them. Many ROM image files will have an extension of a0 or 60 or something like that and it won't be possible to tell if it has a header or not. To determine that look at the file size in bytes. If it's a power of 2 (e.g. 4096, 8192, 16384) then it does not have a header and you can replace the x in the filename with the file letter of the extension (e.g. .a0 becomes .CTa) and if it's a power of 2 plus 2 bytes (e.g. 4098, 8194, 16386) then it has the 2 byte memory address header.

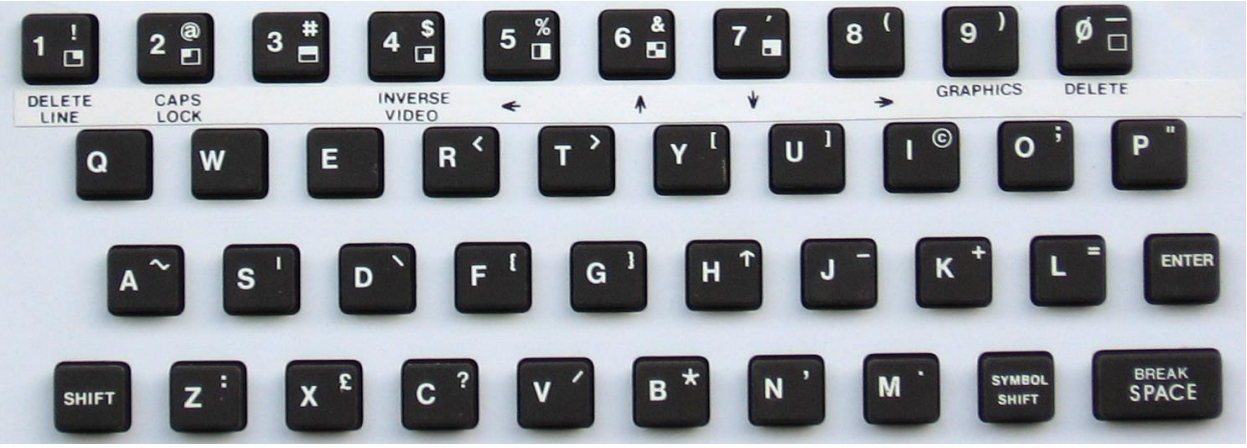
- Load these files via the OSD. There maybe be more than one for a single game… if so load them all.
- From BASIC type `SYS <memory address>` to run the game (or Ctrl+F11 to run at address $A000 which seems to be a common starting address hence the Ctrl+F11 shortcut for `typing SYS 40960`)

Interact Home Computer (1978)

A very early computer selling just a few thousand systems that came with a built in cassette drive and with an original focus on game playing. The MiSTer core supports both .K7 and .CIN files and offers joystick support.

Originally this computer came with Edu-BASIC, but later the Microsoft licensed Level II BASIC was released and offered a substantial upgrade from the original Edu-BASIC.

Jupiter Ace (1982)



The Jupiter Ace is a Z80 based home computer released in 1982. The base configuration consisted of 1K of RAM (although the MiSTer core supports considerably more), 2K of video memory and an 8K ROM with the OS. One major difference between this computer and others from the early 1980s was that instead of the typical BASIC interpreter being the built in language it featured Forth.

Loading existing software is done via the OSD and this core only supports .ACE files.

As you can probably tell, much like many of the computers of this generation, this has a keyboard where each key serves many functions. In addition to supporting Inverse Video and Graphics modes for keyboard entry one can press the Symbol Shift key (which is different from the Shift key) and access the other character on each key (e.g. Symbol Shift + P is the double quote as seen on the upper right of the P key). On a PC keyboard the Control key is the Symbol Shift key.

PC Keyboard Equivalents	
Jupiter Ace	**PC Keyboard**
Symbol Shift	Control
Delete Line	Shift+1
Caps Lock	Shift+2 (Cursor becomes an inverse C)
Inverse Video	Shift+4
Arrow Keys	Shift+5 through Shift+8
Graphics Mode	Shift+9 (Cursor becomes an inverse G)
;	Control+O (as seen on the O key)
“	Control+P (as seen on the P key)

As stated in the introduction of this guide my goal is to get a new user to the point where they can enter a BASIC program that prints HELLO 10 times and load/save their program. Given this computer uses Forth instead of BASIC I’ll try to provide some instructions for doing that in Forth.

First, the prompt at the bottom of the screen is the line editor where you can enter new commands. Try typing “VLIST” (followed by Enter) which is the Forth command for listing all available keywords. Keywords are almost like BASIC subroutines in that they do some sort of task and in Forth you build a

program by creating new keywords that use other keywords that eventually just utilize the primitive keywords that you'll see in this list. As you add new keywords the VLIST command will show those too.

Keywords are created by defining them with the colon (:) operato. Also note that proper spacing is mandatory when defining new keywords in order for things to work (otherwise you'll see an inverse question mark and be invited to correct your mistake). In a nutshell, typing the following will define a new keyword that prints HELLO ten times.

```
: HELLO 10 0 DO CR ." HELLO" LOOP ;
```

and I going to retype that same line with an _ where a space should be just to ensure spacing is done correctly

```
:_HELLO_10_0_DO_CR_."_HELLO"_LOOP_;
```

Also note that " is typed in with a Control+P and the ; at the end is typed in with a Control+O. Once you type that line and press return you will have defined a new keyword named HELLO that prints HELLO ten times. Now you can simply type the keyword "HELLO" and that will happen (just like you can type "VLIST" which is the keyword to list all keywords… in fact after defining the HELLO keyword it should now appear in your VLISTed list of keywords. In that command the CR, ." (period followed by a double quote), DO and LOOP instructions that we used to define our HELLO keyword are all primitive keywords.

You can learn more about the Jupiter Ace and programming in Forth here: http://jupiter-ace.co.uk/usermanual.html.

Unfortunately, this core only supports loading existing .ACE files at present so there is no way to save your Forth programs.

Mattel Aquarius (1983)

Keyboard cursor (directional disc) and F1-F6 keys (buttons 1-6) are mapped to the game controller

Keyboard Mapping

Tab = Toggle between Joystick 1 and 2

.BIN files

Load cartridges (.BIN) via OSD. They automatically start after being selected.

.CAQ files

.CAQ files (most require 16k RAM extension) are usually distributed in pairs. The first part is a BASIC loader and the second part is the machine language portion. To load these:

- From BASIC type `CLOAD`
- Insert the first part of the tape via the OSD
- Wait for it to load
- From BASIC type `RUN` and you should be prompted to insert the cassette and hit Enter
- Insert the second part of the tape via the OSD

BASIC commands

`CLOAD "<optional name>"` = Load cassette program. If a name is given then it will search for that program otherwise it will load the first program it finds (as above).

Notes: The Aquarius used keyboard and controller overlays for its various software titles.

MSX/MSX2/MSX2+/MSX3 (1983)

The MSX is interesting as it's kind of like a PC in that many companies made them based upon a standardized architecture (as provided by Microsoft and ASCII Corp) and it uses a DOS akin to MS-DOS (with many similar commands). It evolved into an MSX2 and eventually an MSX2+ (and MSX3) and this core supports them all. The MSX used ROM cartridges, tapes and floppy disk media for program storage. This system was a major player in Japan, has some great game titles on it and a rich history. There's much to explore here.

Storage

While the MSX supports disk and tape media this core only supports a hard drive image (.VHD file). The default boot.vhd hard disk image will auto load with the core or you can select a different one via the OSD.

You can also mount files on a secondary SD Card although I have not tried that.

Special Keys

F11 = Change CPU speed when in Turbo mode

Core recognizes short (warm) / long (cold) [hold 2+ seconds] reset button (the core specific button on the MiSTer itself)

File Managers

While I'd guess there are other MSX File Managers there are two that are very popular for the MSX called MM (MultiMente) and SF (SofaRun). SofaRun seems to be better at actually running software. One can have their disk, ROM, … etc images all zipped up and SofaRun will handle allowing you to browse inside the zip, decompressing, mounting and loading them.

If all you are after is the games then I'd be doing a disservice if I didn't link to

https://www.youtube.com/watch?v=K5QCWwETnXE&ab_channel=Pezz82

which has a VHD image in the links with pretty much every commercial MSX game out there. Someone did all the work for us.

BASIC

MSX-BASIC is the main version of BASIC for the MSX and there are a few releases of it. You can get a user guide and reference card for MSX-BASIC 2.0 here

https://archive.org/details/AGuideToMSXVersion2.0/A%20Guide%20to%20MSX%20version%202.0/

but to get those of you that know the standard BASIC commands here are a few helpful commands

`CALL SYSTEM` – Exit from BASIC back to DOS
`BASIC` – Return to BASIC from DOS
`SAVE "[Drive][Filename]"` – Save a BASIC program to disk (e.g. `SAVE "A:TEST.BAS"`)
`LOAD "[Drive][Filename]"` – Load a BASIC program from disk (e.g. `LOAD "A:TEST.BAS"`)

NEC PC-88 (1981)

The PC-88 is a popular Japanese based home computer based on a Z80 compatible processor running at 4 MHz and supporting up to 64KB of RAM. Many variations of this machine were released sporting support for different graphics modes and audio capabilities. N88-BASIC is built in to the system.

Controls

Often Numpad 8/4/6/2 or cursor keys and Enter/Space/Shift/Esc/Tab/Z/X/C/V.

CPU Speed of 8 MHz is probably ok except for some older games that might run too fast. You'll need to be in the correct graphics mode.

Commands

```
FILES
LOAD"filename"
RUN "Filename"
MON
R
RUN
```

Orao (1984)

The Orao core supports only cassette tape based software.

.TAP/.WAV Files

- After powering on the Orao type `BC` to enter BASIC
- Press the Enter key to bypass the MEM SIZE prompt
- Type `LMEM ""` to load the program
- Select the file you want to load via the OSD
- As the program is loaded status updates may display on screen

The MiSTer github actually has some very good documentation for this computer.

Oric-1 / Oric Atmos (1982)

Oric Atmos keyboard

The Oric Atmos was the successor to the Oric-1 and added an improved keyboard and an updated ROM.

Keyboard Mapping

F10 = NMI key (Warm start)
F11 = Reset (Cold start like turning the computer off and on. Use to reboot after a .DSK image is selected via the OSD)

.DSK files (disk images)

- Select bootable .DSK file from the OSD
- From BASIC press F11 to reboot and load the bootable disk

Notes

The Oric core supports .DSK images however there are several .DSK image formats behind that file extension. This core only supports Amstrad CPC Extended Format disk images (and there are a few samples in the MiSTer github repository for this core). This is unfortunate as the TOSEC set is mostly (if not all) in MFM format (and you'll see an 'insert system disc' message if you try to boot with one of these). Conversion can be done and there are instructions as to how to do that in the github repo for doing so one at a time.

Once converted some of these disks issue a warning that the DOS has been tampered with. I had to boot from a SEDORIC 3 disk. It would boot to a menu that let me exit to BASIC. I would then swap in the disk with the game I wanted to test. After doing so I could type DIR to see the files on the disk and then the name of the program and it would load.

After this success I was motivated to see what I could do about that via either creating a batch conversion utility or trying to add support the MFM .DSK format as I was able to track down specifications for both file formats. However, after sampling some conversions and not having a lot of luck using this core (text rows on the screen would be offset incorrectly with words wrapping and I could

not find any keyboard keys that would control a given game) I'm waiting until I either figure something out that I don't understand or updates are made. I welcome any feedback.

There is an Oric DOS (Disk Operating System) that was the original DOS for the Oric, but it appears another DOS named SEDORIC had become the standard and is actively supported with a recent 4.0 release.

PC 486DX 33 [AO486 core] (1990)

Win+F12 = OSD (as F12 key is the PC's F12 key)

Simulates a 486DX 33 MHz PC (no floating point co-processor) with SVGA, SoundBlaster 16/Pro, MIDI, Hard Disk (via .VHD files) and CD-ROM support. As PC keyboards are pretty standard and I'd guess many use one with their MiSTer I have not included a picture of one as the keyboard mapping is pretty close to 1:1.

Operating Systems

A PC can use many Operating Systems, but MS-DOS/Windows were the most popular back in the early 1990s when this Intel CPU was released. MS-DOS (Microsoft DOS) has several version releases, but 6.22 was the common DOS at this point in time (along with Windows 3.1 which sat on top of DOS). Later Windows '95 came along to supplant DOS entirely and is still in spec, but Windows '98 required a 486DX 66 minimum although it will still function on this core (and spec is hard to quantify in this core as the clock can be as high as 90MHz so 33MHz is just a performance estimate).

Getting Started

Because of this core's theoretical flexibility and the fact that you can go in many different directions with it it's hard to cover all the possibilities. I'll just say that I was able to use a set of MS-DOS 6.22 install disks with a virtual hard drive image to install DOS on the hard drive. I wanted to install Windows 3.1, but you'll need a CD-ROM device driver for MS-DOS and it appears that is among the several utility programs available for this on the AO486 github page (there also a device driver that gives you direct access to files on your MiSTer's storage device). I was also able to do the same and install Windows '95 for a CD image onto the hard drive and while it ran the first time it failed to start the next time I tried to boot from that hard drive image.

BASIC

QBASIC is the BASIC that came with MS-DOS 6.22. It offers a text based GUI and I was able to type in and run my sample BASIC program. It offers a text based GUI. Access menus there by holding down the Alt key.

.IMG files

MiSTer uses .IMG files as virtual floppy disks and .IMG files are about as simple a format as can be. They are just a dump of the raw sector data from the disk. A standard 3.5" double sided, high density disk image will be 1,474,560 bytes in length (2 sides * 80 tracks * 18 sectors per track * 512 bytes per sector).

.VHD files

PCs from around 1990 usually had a ~200 megabyte hard drive and one would install DOS and games to it from 3.5" and 5.25" floppies. Start by creating a Virtual Hard Drive (.VHD) file (You can use the Windows 10 Disk Management Utility to do this) and install MS-DOS 6.22 onto it. You'll need the MS-DOS 6.22 3 disk .IMG file set and you'll need to start by ensuring the floppy drive boots first, the virtual hard disk is mounted, and MS-DOS 6.22 disk 1 is in the floppy drive. Upon booting the MS-DOS 6.22

install program should load and allow you to format your virtual hard disk and install/configure MS-DOS 6.22. From there you'll need disk IMG files to install.

PCs of the day had a bit of a memory management issue due to the initial 640KB limitation on the IBM PC (although the 8088 had 20 address lines which meant it could address up to 1 megabyte). Different drivers such as HIMEM.SYS and EMM386.EXE because a part of later versions of DOS (e.g. MS-DOS 6.22) for accessing larger amounts of memory and eventually memory beyond 1 megabyte.

.CHD/.IMG/.BIN & .CUE files

These are CD-ROM images (note that the .IMG extension is also used for floppy disks) which can be mounted via the OSD.

PDP-11 [DEC] (1970)

Upon booting this core the game Spacewar! is automatically loaded.

Keyboard Mapping

F1 = Power on/off
F2 = Toggle active switch row on the console view. This positions the cursor on the first switch in the row. Move left-right with numeric keyboard arrows and toggle switch with enter.
F3 = Single instruction switch toggle
F4 = Cycle between CRT output, Console output and Teletype.
F5 = Start
F6 = Stop
F7 = Continue
F8 = Examine
F9 = Deposit
F10 = Enable read-in mode
F11 = Tape feed

Loading a .RIM program file:

Press F10 to enable read-in mode (or select Enable RIM mode in the OSD)
Use OSD to select .RIM file

SAM Coupé (1989)

Keyboard Mapping

F11 = NMI key
Ctrl+F11 = Reset
Alt+F11 = Reset and unload disk images

Loading a .DSK image is done via the OSD and then it will auto boot

Sinclair QL (1984)

The Sinclair QL was a 68k based, high-end follow up to the Sinclair ZX Spectrum and was meant for more of a professional user. It used microdrives for storage which used magnetic tape and were meant to be more cost effective versus floppy disks however reliability got the better of that and soon floppy disk interfaces because available and that became the more preferred method of storage.

QDOS (System Firmware)

The main github repo is a bit out of date with respect to some of the ROM files so you'll probably want to visit https://www.kilgus.net/ql/mister/ and get the QL OS zip file there. It has both the 'original' Sinclair QDOS ROM and an enhanced Minerva based ROM. The Minerva based ROM is a substantially upgraded ROM that is (seems to be?) compatible with all of the QL's software. You can experiment with both and, for a start, see just how much faster the Minerva based ROM boots. Furthermore, these images have been enhanced with QL-SD which is an SD card interface for the Sinclair QL allowing you to load software for the MiSTer I/O board's SD card as well as .WIN files. There's also a sample QL-SD.WIN file here.

If you really want to have a great time with the QL I suggest pulling down the Sinclair QL User Guide from https://archive.org/details/sinclair-ql-user-guide., but I'll walk through the basics here. The first thing that happens when booting your QL is you'll be asked to press F1 if you're using a monitor and F2 if you're using a TV (presumably because F2 switches to a lower resolution graphics mode that a TV can handle?). I've been pressing F1. Your screen will then be divided into three areas with the bottom area allowing you to type immediate commands or enter BASIC program line by line in Super BASIC, the top right displaying output, and the top left allowing you to list the lines in your current BASIC program.

Loading Programs

You can mount.MDV (microdrive) images (a format for Microdrive cartridge preservation used by the old QLAY emulator) or .WIN (SD card) images from the MiSTer OSD. You can get a list of files on the device with the 'dir' command however you must specify the device you want a directory from and device names are odd as they include a trailing underscore character:

`mdv1_` and `mdv2_` are the two Microdrives that this core supports
`win1_` is the .WIN file in your QL directory on the MiSTer SD card
`win2_` is the extra SD card in the I/O board's slot (I have not tried this yet)

so

`dir mdv1_` will list files on the .MDV file in simulated Microdrive #1 (and it'll be just as slow as a real Microdrive so be patient)
`dir win1_` will list files on the .WIN file in the simulated QL SD interface (much faster than a Microdrive)

Often there will be a file named `boot` and you can load and run these using the `lrun` command by specifying a name that includes the device the file is on and the filename.

`lrun mdv1_boot` will load and run the program boot on Microdrive #1.

You can also load and save Smart BASIC programs to a device with the load and save commands

`save mdv1_test` saves the BASIC program in memory to a file names test on Microdrive #1.
`load mdv1_test` loads the BASIC program test on Microdrive #1 into memory.

Some helpful key commands include

Ctrl-Space = Stop the running BASIC program
Ctrl-Left Arrow = Backspace
Ctrl+Right Arrow = Remove the character to the right of the cursor

Transferring files from a PC to the QL core

Unfortunately I have not found a great way to do this yet. I have yet to experiment with the SD card in the I/O board, but I read that files must be stored sequentially (no fragmentation) so that doesn't seem like a great thing and I can't find current tools to work with .MDV or .WIN files. There is a tool called qxltools for working with these .WIN files, but there's no current Windows version of it.

Conclusion

This seems like a pretty neat and powerful (for the time) computer with a lot to it. I've been able to play some not so great games on it, get what seems like a GUI going from the QXL.WIN sample image with mouse support, and it seems like there's quite a bit more to learn. Hopefully this gives you the start I didn't have so you can see that the core itself it pretty great!

Sinclair ZX Spectrum (1982)

The Spectrum was a dominant computer in the United Kingdom with several iterations that are all supported by this core. It has a sizable software library and quite a following even today with the release of the reimagined ZX Spectrum Next (a core for that is also available).

Depending on the model of Spectrum you are using upon booting the system you may see a menu with options like Tape Loader, 48 BASIC and/or TR-DOS or you might just see a plain "© 1982 Sinclair Research Ltd" on an earlier ZX Spectrum 16KB/48KB. Select 48 BASIC from the menu or just press Enter from the "© 1982…" screen to get to BASIC.

The Spectrum didn't come with any joystick/gamepad controllers so many games support keyboard controls, but the Kempston joystick seems to be a common choice for game players. Kempston is an interface to allow DE-9 (Atari) style joysticks to be used with the Spectrum.

From BASIC

The cursor will show a K initially indicating that a keyword is expected. If you were to press P, for example, the BASIC PRINT statement would be auto-typed (note that PRINT is on the P key) instead of just P. The cursor will then change to an L indicating it is expecting a (lowercase) letter. The character in the cursor tells you how the next keypress will be interpreted with K meaning keyword, L meaning lowercase, C meaning capital and E meaning extended mode. The Control key on a PC keyboard is used as the Symbol Shift key (lower right) of the Spectrum. When entering a BASIC command the first part will be a [K]keyword and then the cursor will change to an L. You can enter [E]xtended mode by pressing Ctrl+Shift on the PC keyboard which will change the cursor to an E and that will give you access to the Green keywords. The red keyword below the keys are also accessible from Extended mode by holding Symbol Shift (the PC's Ctrl key) down when pressing the key you want after entering Extended mode. Note that some of the extra PC keys have been mapped to Spectrum keys such as the PC's single/double quote key although it's flipped, but that means you can type a double quote by press the PC keyboard's single quote key OR via Ctrl(Symbol Shift)+P.

.TAP/.CSW/.TZX tapes

- From BASIC type `LOAD ""` (via these three keys: J Symbol Shift+P Symbol Shift+P or J ' ' on a PC Keyboard) or from the System Menu choose the Tape Loader
- Select tape file from the OSD
- Once it loads (which may take a long time) you may also need to type `RUN` from BASIC

.DSK disk images

Note: You must be in Spectrum+3 mode (which means Spectrum + a 3" disk drive) choses via the OSD under Hardware->Memory.

- Booting the Spectrum in Spectrum+3 mode means Loader will be a menu option
- Choose the .DSK image via the OSD
- Press Enter with the Loader menu option selected and the disk will boot

.TRD TR-DOS disk images

Note: You must be in a mode that supports TR-DOS so that it appears on the system menu upon booting the computer.

- Enter TR-DOS from the system menu
- Select a disk image from the OSD
- From TR-DOS press R (to autotype the RUN command) and press Enter

.Z80/.SNA snapshots

- Select file via the OSD and it should automatically run. Easy enough.

Keyboard Mapping

F1 – pause/continue loading
F2/F3 – Jump to previous/next part of tape during load
F4-F8 – Toggle CPU speed between 3.5/7/14/28/56 MHz
F9 – Pause/resume
F10 – Switch to 48k BASIC and automatically `LOAD ""` a tape image
Right Shift+F10 – Same as F10 with 48K lock (?)
F11 – Enter +D (?) snapshot menu

    3 – Screenshot
    4 – 48k snapshot
    5 – 128k snapshot

Ctrl+F11 – Reset (warm start)
Alt+F11 – Reset (cold start)
Ctrl+Alt+F11 – Reset to ROM0 menu
Right Shift+F11 – Enter Multiface 128 menu

Alt+F1 – Alt+F6 – Quickly choose between (F1) ZX Spectrum 48k/ (F2) Spectrum 128K/ (F3) Spectrum +3/ (F4) Pentagon 48k/ (F5) Pentagon 128k/ (F6) Pentagon 1024k

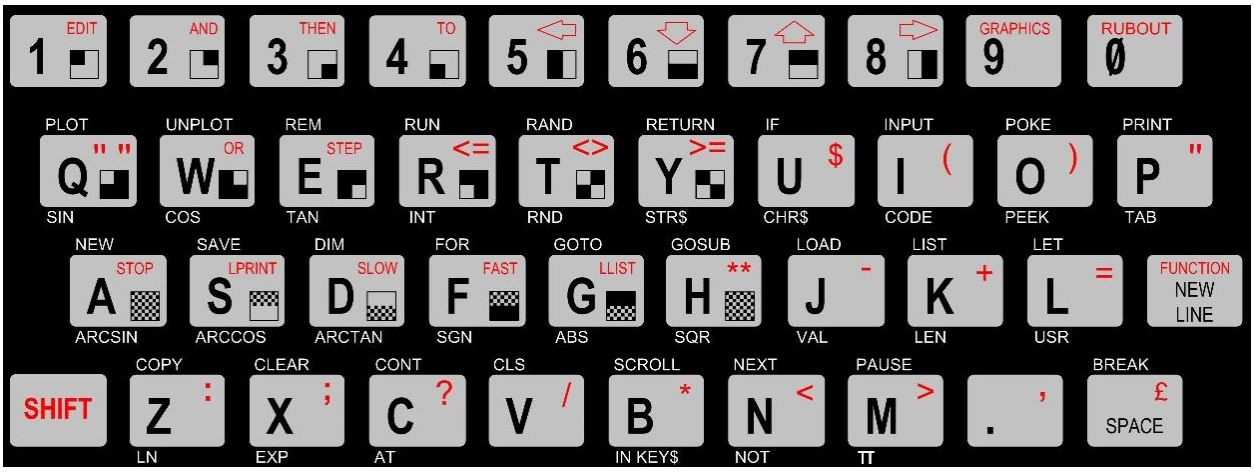
BASIC commands

`CAT 1` – List contents of .IMG/.MGT disk image snapshot (I have not found any of these to try yet so I can't comment further on them)

Sinclair ZX80 [Sinclair ZX81 Core] (1980)

The ZX81 core also simulates the ZX80 and that can be chosen via the OSD. The ZX80 uses .O files for its software images.

Sinclair ZX81 (1981)



The ZX81 came with 1K of RAM and was expandable to 16K with a ‘memory pak’ that plugged into the back. This core allows for using larger amounts of memory than that. In addition to this expansion port the ZX81 had ports for power, video, and a cassette recorder. It didn’t even have a power switch; you would just plug/unplug the power cable to turn it on/off.

Note: This computer was released as the Timex Sinclair 1000 in North America. The main differences between the two was that the TS1000 had 2K of memory in the base unit and some renaming of keyboard keys such as ‘Enter’ replacing ‘New Line’ and ‘Delete’ replacing ‘Rubout’. Later the Timex Sinclair 1500 was released with an upgraded chicklet keyword and a full 16K of RAM.

Startup

After loading this core it will boot to a screen that is blank with the exception of an inverse video ‘K’ in the bottom left. You’re in the ZX81’s BASIC and the K means it’s expecting you to type in a keyword. If you look at the ZX81 keyboard (pictured above) you’ll see BASIC keywords littered all over it with most keys having 4-5 different functions. BASIC keywords are listed above the keys so if you press the letter P (while in [K]eyword mode based on the cursor) you’ll actually see the BASIC keyword ‘PRINT’ autotyped AND the cursor will change to an L indicating that it is now expecting a letter.

If an inverse S appears after entering your BASIC command that indicates a syntax error… the ZX81 doesn’t recognize the command you typed near where the inverse S appears.

The Shift key is used to access the red characters / keywords on each key hence it being red in color.

The New Line key is the PCs Enter key (which is why it was renamed for the North America release).

The commands listed below the keys are functions and are accessed by pressing Shift+Enter on a PC keyboard (accessing the red FUNCTION keyword on the ZX81’s New Line key). The cursor will change to an inverse F when it is expecting a function keyword.

The graphic symbols (or characters) on the keys are accessed via entering Graphics mode (Shift+9) which will change the cursor to an inverse G until you again press Shift+9 to exit keyboard graphics mode.

If you type a line number before your command then the line of code will be added to your program (type a line number with nothing after it to remove a line of code) and you will see your current program listing above the bottom area where code is entered. Type the RUN command via pressing the R key to run your program. Once it complete it will display #/# and you'll need to press New Line (Enter) again to return to entering BASIC commands.

For more info on using BASIC on a ZX81 please read the book ZX81 BASIC Programming that came with the ZX81:

http://zxnext.narod.ru/manuals/ZX81_Manual.pdf

Unfortunately I don't see a way to save a program here so other than playing around I am not sure there's much point.

.P Files

.P files are digital versions of the analog audio that would be played from a ZX81 program's cassette tape. This core supports loading these programs from BASIC:

- Using the OSD select the .P file you want to load
- From the BASIC K prompt type `LOAD ""` (J, Shift+P, Shift+P on your PC keyboard)
- You may have to press R to autotype `RUN` to run the program

Keyboard Mapping

Given how many more keys a PC keyboard has there are several keys that are used to provide shortcuts to ZX81 keys:

‘ = “ (so you can press ‘ instead of Shift+P for a double quote)
Backspace = Rubout (great because it'd drive me nuts having to press Shift+0 for this)

Also note that this core has options for a gamepad (joystick) controller so that it can act like the cursor keys on the keyboard (among other options). Nice!

Note: The github repo for this core recommends using 16K of Main RAM, 8K of Low RAM, CHR$128/UDG=128 chars, QS CHRS=enabled and CHROMA81=enabled for running most games. UDG is an expansion board that allowed the ZX81's character set to be redefined and CHROMA81 allowed for color graphics on the ZX81.

Spectravideo SV-328 (1983)

The SV-328 is a Z80 based computer that came with 64KB of memory and Microsoft Extended BASIC. It had much in common with the MSX in terms of hardware, but was not compatible. Upon startup it will boot to BASIC. This core supports cartridge and cassette based software and the system itself supported a wide range of peripherals (such as disk drives) that are not currently supported by this core.

Using Cartridge based software

Select the cartridge .ROM file from the OSD and it will automatically start. [At present you'll need to reboot the MiSTer in order to use a different cartridge after selecting one.]

Loading a BASIC program

- Select the .CAS file from the OSD and then exit the OSD
- Optionally enter the BASIC command `sound on` if you want to hear your cassette load (it can take a while so sometimes it's nice to have some audio feedback that something is happening)
- From BASIC type `cload`
- Once the program loads you will need to type `run` to execute the program

Notes:

`cload` can also be used with a filename to find a specific program on a cassette e.g. `cload "game"`. Filenames consist of 6 or fewer characters. There is also a `cload?` Command which is used to verify that a program is correctly stored on a cassette. Finally, if saving to cassette was supported, there is a `csave` command.

Loading a Machine Language program

- Select the .CAS file from the OSD and then exit the OSD
- Optionally enter the BASIC command `sound on` if you want to hear your cassette load
- From BASIC type `bload "cas:",r`

- The program will load and automatically run

Notes: The general format of the bload command is `bload "<device><filename>"[,r]` where, in our case, the device is the cassette drive, we are not searching for a specific file (just the first one on the cassette), and the ",r" mean to automatically run the program after it loads. If the core supported disk drives then the device name would be a "1:" for the first disk drive and, yes, there is a corresponding `bsave` command.

Also note that some machine language programs have a BASIC loader so you'll need to cload the BASIC loader program, type run to run it, and then it will proceed with loading the machine language portion of the program

User Manual

https://archive.org/details/svi328usermanual

BASIC

The BASIC interpreter on this computer is line number based so my sample BASIC program runs fine here. Note that you can type in BASIC commands in upper or lower case. BASIC also offers 5 'shortcut' commands displayed along the bottom of the screen that are accessed via F1 – F5 keys.

While you can type in a BASIC program there does not appear to be a way to save your work (I have not tried the ADC converter with a real cassette). I was able to find this manual for SVI-328 BASIC:

https://archive.org/details/svi328basicreferencemanual

TI-99/4a [Texas Instruments] (1981)

This core initially loads to a blank screen and you'll need to have a proper BIOS ROM in order for the core to do anything (preferably one with the disk ROM code now that this core support floppy disk images). If that is present, it will boot to the TI Welcome screen which will give way to the TI menu which will contain a "1 FOR TI BASIC" option. Press 1 to enter TI BASIC. If you want to exit BASIC and return to the menu then type the `BYE` command from TI BASIC.

TI-99/4a ROM Files

ROM files often come as multiple files such that each part of the ROM has a name followed by a C, D or G followed by a .BIN extension. In order to load a ROM just load each part of the same ROM one by one. [Yes, you could load parts of different ROMs, but needless to say that won't end well] After each file is loaded the core returns to the TI Welcome Screen and Menu and now you should have the additional boot menu options for running your ROM in addition to TI BASIC.

Some ROM files do have all the pieces merged into one and in that can you can use the load full ROM option from the OSD.

Keyboard Mapping

The FCTN (function) key (lower right) is simulated by the Alt key on a PC keyboard and is used to access functions on the gray bar above the number keys (hence the colored dot to the right of the line and on the FCTN key) and the symbols on the front of some alphabetic keys. For example, if you enter the TI's BASIC then Alt+= (Alt+- on a PC keyboard) is Quit (return to the boot menu) and Alt+3 is Erase (Backspace).

Other keys on the keyboard are mapped to a PC keyboard simply based on the shape of the TI keyboard so

TI Key	PC Key
/	[
=	-

Floppy Disk Support

Cassette tapes aren't currently supported, but single density (single and double sided) floppy disks are. There is a utility called TI99Dir (https://www.ti99-geek.nl/ under Projects) that will allow you to create blank disk images. Once you create a proper disk image you can mount it and load/save programs on it. The TI BASIC commands for doing so are

`SAVE DSK1.<FILENAME>` = Save BASIC program to the disk in drive 1 as file <FILENAME>
`OLD DSK1.<FILENAME>` = Load BASIC program from the disk in drive 1 named <FILENAME> from disk

Note that DSK1 and can replaced with DSK2 for disk drive #2.
Also note, I suspect OLD is used instead of LOAD because it's the opposite of NEW. NEW clears out a BASIC program and OLD restores it. Obviously someone at TI thought that'd be a great name for the load command.

One of the challenges with the TI is things don't seem to be very integrated. Do you want to know what files are on a disk? I guess you were supposed to write that down because for that you need to use the TI Disk Manager ROM pack. It provides basic disk facilities such as listing the files on a disk along with their types:

`PROGRAM` = Memory Dump (might be BASIC or Machine Code)
`DIS/FIX 80` = Relocatable Machine Code
`DIS/FIX 163` = Merge Files (Extended BASIC)

so working the other way, if you were to find a TI disk image and get a list of files on it then files of type PROGRAM might be BASIC programs that could be loaded as above OR they might be EA files (Editor/Assembler) and you'll need the Editor/Assembler ROM pack to loaded them using option 5.

DIS/FIX 80 files are also loaded with the Editor/Assembled ROM pack, but you'll use Option 3 for those.

Extender BASIC (XB) supports some basic disk functionality built in such as

`CALL DIR(1)` = List the content of files on disk 1 if the disk controller supports it (otherwise you'll need to use a BASIC program or the Disk Manager ROM cartridge to view files on disk)

Cassette Support (if cassette tape support is added)

`SAVE CS1` = Save BASIC program to cassette
`OLD CS1` = Load BASIC program from cassette

https://www.ninerpedia.org/wiki/TI-99/4A_system_usage

TRS-80 Color Computer 2 [Tandy] (1983)

The TRS-80 Color Computer 2 is the fully compatible successor to the TRS-80 Color Computer (released in 1980) and featured a smaller case, a 'real' keyboard vs the chicklet style of the original, and more memory in its base model.

Keyboard Mapping

End = Cold start
F9 = Run a cartridge game (aka Program Pak) after it has been inserted via the OSD

After inserting a disk via the OSD you can use the following commands to see what's on the disk and run programs from disk

`DRIVE #` = Set the default drive to drive #
`DIR #` = Disk directory of drive # where # is optional

Running a BASIC program (BAS extension in the DIR):

`RUN"NAME` = Run the BASIC program (.BAS extension) from the disk

Running a Binary program (BIN extension in the DIR):

`LOADM"NAME` = Load BINARY (.BIN extension) file
`EXEC` = Execute the loaded binary file in memory

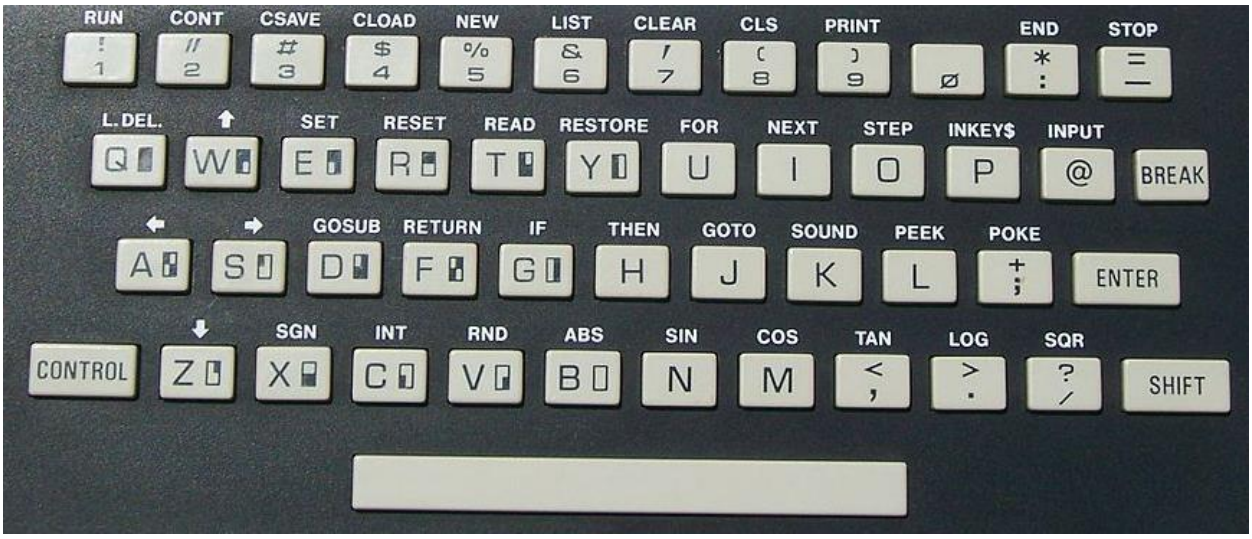
Booting a disk that DIR doesn't display a directory for:

`DOS` = Command to boot an OS/9 disk

TRS-80 Color Computer 3 [Tandy] (1986)

Info for the TRS-80 Color Computer 2 applies here as this is fully backwards compatible. Additional info coming soon…

TRS-80 MC-10 [Alice] (1983)



Each key has a shortcut listed above it which is accessed via the Control key. The core supports the 16k RAM expansion and cassette drive via both .C10 files and the MiSTer ADC add on board (real cassettes).

.C10 files (loading cassette programs)

- Boot into the core (Microsoft BASIC)
- In the OSD select the cassette game you want to load.
- From BASIC press Control+4 which will display the CLOAD command on screen. Press Enter.
- From the OSD click on 'rewind' and then click on the 'play' option.
- You should see a flashing S (searching) in the upper left corner that changes to an F (found) along with the name of the program that is loading.
- Once the game loads just type `RUN`.

Note: I have had minimal success with this… some games don't seem to load properly and others just don't run. In many cases I can type the BASIC LIST command to see what BASIC code loaded into memory and usually that looks all right although sometimes it does not. Having never owned one of these and not being sure what to expect I can't diagnose where things are going wrong. I tried enabling/disabling the 16k memory expansion in my trials. The notes for this core indicate an existing issue with switching video modes so hopefully that's the bulk of the issue I am experiencing.

TRS-80 Model I (1977)

The TRS-80 Model I core has grown to become quite full featured. Note that on the keyboard the Clear key (next to the white Enter key) is simulated by the PC keyboard's Home key and Break (upper right) is the PC keyboard's Escape key.

When booting you will either need to insert a disk with a DOS (Disk Operating System) on it OR quickly press Escape to start the system as a cassette based TRS-80. Note that many cassette games won't work on a DOS (disk) based system.

To load a machine language cassette program:

After the core boots and you quickly hit Escape you'll see a 'READY?' Prompt
Press Enter/Return and, after a second or two, you'll see the 'READY' prompt shift to the bottom of the screen
Type `SYSTEM` and a '*?' prompt should appear
Go to OSD to select cassette to load into cassette drive
Type the six letter name (the first letter will suffice) of the file to load, press Enter and wait for it to load
Type `/` to run the game

To load a BASIC cassette program:

`CLOAD` = Load cassette game
`RUN` = After CLOADed to run loaded cassette game

In order to load disk based software you will need to boot with a DOS. The following DOS Commands will be helpful:

`DIR #` = Show a disk directory when # is the drive number (drives 0 and 1 are supported in this core)
<filename of CMD file> = Load and run program (CMD files are machine language programs)
`BASIC RUN"<filename>/BAS"` = Init BASIC and run BASIC program (BAS files are BASIC programs)

JV1/JV3/DMK .DSK images

JK1, JV3 and DMK a formats of .DSK image file. This core only supports JV1 format .DSK files.

X68000 [Sharp] (1987)

The Sharp X68000 is a Japanese computer that is renowned for its great arcade game ports from the late 80s and early 90s. It had many different models with gradually upgraded specifications through the years. The MiSTer core offer two CPU speeds and a faithful reproduction of this classic computer.

The core supports two floppy drives and one hard drive and uses .D88 floppy disk and .HDF hard disk images. Many of the floppy disk images I have found are in the .DIM format and need to be converted to .D88 for use with the MiSTer core. There is a Python based command line utility here (http://www.formauri.es/personal/pgimeno/misc/converters.php) that will accomplish that task although you'll need to install Python to do that. The format of the Python command would be

```
python dim2d88.py <.dim input filename> <.d88 output filename>
```

The main operating system for the X68000 is Human68k which is quite similar to DOS. Many concepts such as the A: and B: drives being the two floppy drives and C: being the primary hard drive are the same as are commands such as those that follow. Filenames consist of 18 (not 8 like DOS) characters and a 3 character extension with the extension 'X' being used for executable programs. I read somewhere that paths are separated with '/' as Unix uses, but it seems like '\' works for me when I load and save BASIC programs. Manuals for the X68000 are readily available online, but they are in Japanese and of little use to me and make figuring out how to do things with this computer that much more difficult.

Games

There are lots of great games available for this machine and many of those that I have found are on auto boot disks (floppy and hard drive images)… just mount the disk via the OSD and reboot (which seems to automatically happen for .HDF images). The X68000 used an MSX style gamepad controller (which was similar to an NES controller with a directional pad, 2 buttons, and 2 more buttons for starting the game).

DOS Commands

`DIR [optional drive/path]` = List files in a directory on a disk (note that some message in Japanese seems to appear during first access to a disk… I presume that's just letting you know you changed disks… press A to continue).

`MORE <[filename]` = View the contents of a [text] file on the screen.

`EDIT <filename>` = Simple text editor although I have yet to figure out how to load/save you can exit it press pressing Escape, Q and Y (which I presume Escape enters some sort of command mode, Q is the Quit command and Y confirms that we really want to quit).

DOS Command Line Interface (CLI)

Some of these (such as MORE) are commands contained in the BIN directory on the Human68K boot disk and therefore won't work if that disk is removed. After changing disks and accessing a new disk for the first time I see a message in Japanese in the middle of the screen and responds to the A, I and R keys… I'm guessing this might be akin to the DOS "Abort, Retry or Fail?" message. If I press A after issuing a DIR command and seeing this message the next DIR command seems to work… likewise R seems to give me the directory listing right away.

There are also some function key shortcuts listed at the bottom of the screen. I can tell F1 (C1) repeats the last entered command a character at a time and F3 (CA) repeats the entire last command, but have not figured out exactly what the others do.

BASIC Programs

The X68000 uses a BASIC interpreter called X-BASIC which is included on the Human68K boot disk in a folder named BASIC2. From within X-BASIC, various shortcut commands such as LIST for listing out the lines in the BASIC program in memory are available as shortcut keys listed at the bottom of the screen and accessible with the F1-F10 keys. I was able to type in a short program, save and then reload it.

BASIC Commands

`LOAD "<filename>"` e.g. `LOAD "A:\FOLDER\FILE.BAS"` = Load the BASIC program named FILE.BAS in the folder named FOLDER on the disk in the first floppy drive (A:).

`SAVE "<filename>"` e.g. `SAVE "B:\FOLDER\FILE.BAS"` = Save the BASIC program in memory to a file named FILE.BAS in the folder named FOLDER in the root directory of the floppy disk in the second floppy drive.

`SYSTEM` = Exit BASIC and return to the DOS CLI.

SRAM

The X68000 uses SRAM to store system settings (I think the Human68K program named SWITCH allows you to set things up) and small programs. I am unclear as to exactly how this all works with the MiSTer core.

ZX Spectrum Next (2017)

The ZX Spectrum Next is a fan made, souped up ZX Spectrum computer and comes with (on the github site) a bootable hard drive image that contains various programs. The ZX Spectrum Next is, itself, FPGA based just like the MiSTer.

F1 = Reset
F3 = Toggle 50/60Hz
F8 = Change CPU Speed (3.5/7/14/28 MHz)
F10 = DivMMC NMI
F11 = NMI/Multiface

Adventure Vision [Entex] (1982)

The Adventure Visions is a cartridge based console that had a built in screen made of red LEDs in a 150 by 40 arrangement. Only 4 games were ever commercially released for the system as the system did not sell many units however its unique nature makes it a bit of a collectors item. Controls consisted of a joystick and 4 buttons labelled 1 through 4. One set of buttons was on the left and another on the right for ease of play by both right and left handed people.

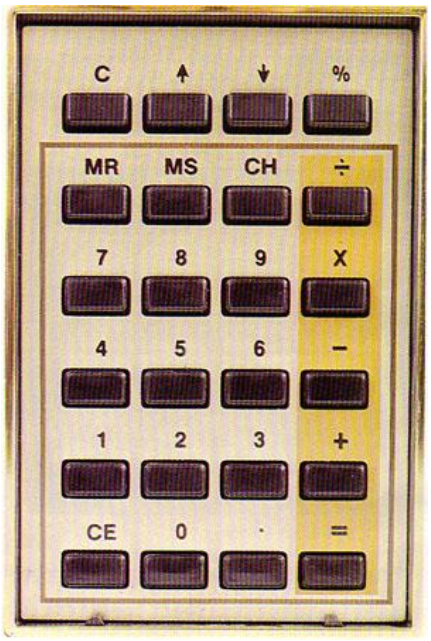
Arcadia 2001 [Emmerson] (1982)

The Arcadia 2001 was released as a budget system and came with two Intellivision style controllers that had 12 button keypads, a joystick and fire buttons on the side. It wasn't a commercial success and had only about 50 games released for it during its short lifetime. There also many clone systems including the Bandai Arcadia in Japan.

Astrocade [Bally] (1978)

The Astrocade console has a 4 by 6 (24 key) calculator like keypad on the console and game cartridges that have a form factor very similar to cassette tapes all the way down to an eject button on the console for them. Its controllers were certainly unique. They had a trigger style fire button with a combination joystick/paddle on top that you could operate with your thumb in joystick mode. In an attempt to be more educational the Astrocade included Astro BASIC. Given the small keypad you can imagine what a delight it would have been trying to enter a program. Furthermore, the BASIC cartridge itself had an audio connector to attach a cassette recorder to the system to load and save programs.

By Evan-Amos - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=18312472



Atari 2600 (1977)

The Atari 2600, while not being the first cartridge based game console, is often cited as the game console that started it all. The console itself had six switches (Power, Color / Black & White, two [left and right] Difficulty switches, a game Select switch [as games often came with many variants] and a game Reset [aka Start] switch). I know of at least one game that used these switches as part of its controls (Star Master). By default Select and Reset are mapped to the gamepad, but Color/B&W and Difficulty are in the OSD. You can remap all of these via the OSD.

The joysticks were simple with a single button and the console originally came bundled with two joysticks and a pair of paddle (spinner) controls. The joysticks became an industry standard using a DE-9 style connector that was used on many other computers and consoles that followed. The paddles did have a limit as to how far they could be turned (the driving controllers were similar to the paddles, but did not have the limit). They also, later, released a couple keypad style controllers for games like Codebreaker and Star Raiders.

Note: This system is now supported via the Atari 7800 core. The Atari 7800 system had built in Atari 2600 compatibility and so does that core.

Atari 4 way/1 button joystick controller

Atari paddle/spinner with 1 button

Atari 5200 (1982)

This is the same core as the Atari 800XL core as the two systems had almost identical hardware. The biggest difference may have been the controllers: the Atari 8-bit computers had a keyboard and traditional one button joysticks/paddles, but the Atari 5200 had an analog joystick with 2 buttons at the top of each side, Start/Pause/Reset buttons along the top, and a 12 key keypad with the digits 0-9 * and #.

Atari 5200 Controller

Start/Pause/Reset across the top

2 buttons on each side

Analog joystick

12 Key Keypad

Atari 7800 (1986)

By Evan-Amos - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=18312472

The Atari 7800 was a kind of successor to the Atari 2600 in that it featured backwards compatibility with Atari 2600 games (as does this core). Furthermore, unlike the Atari 5200, Atari didn't go nuts with the controller and just added an extra button and changed it to have more of a grip style.

Atari Lynx (1989)

The Atari Lynx was Atari's technically superior, portable Game Boy competitor featuring color graphics and advanced scaling hardware to provide. It also consumed batteries very quickly making an AC adaptor an important add-on diminishing its portability. The console features a DPad and a two buttons labelled A and B for game playing. It had two of each button so left handed users could flip the console over and play in comfort. Additionally it had two option buttons, a restart button and a flip button (for the aforementioned feature).

The core provides 4 game save states accessed via F1-F4 to load and Alt+F1-F4 to save.

ColecoVision / Sega SG-1000 (1982)

The ColecoVision was every 1982 arcade aficionado's dream come true boasting (and delivering) an 'arcade at home' experience and including the immensely popular Donkey Kong as a pack in game. Its controllers were geared at being held by one hand that could squeeze the two buttons on the left/right side of the joystick along with a 12 key keypad. Some games would include overlays for this keypad (much like on the Intellivision). The system also offered an expansion slot on the front right that supported an Atari 2600 adapter, a roller controller (track ball) and a driving controller (for games like Turbo). An advanced joystick type controller was also later released called the Super Action Controller.

This core supports .COL/.BIN/.ROM game cartridge ROM files.

Fairchild Channel F [Fairchild] (1976)

The Fairchild Channel F was among the pioneering systems of the age of ROM cartridge based video game consoles. The main console had 5 buttons on the front left (RESET and numbers 1-4). The buttons are labelled

RESET	TIME 2 MIN **1** HOCKEY	MODE 5 MIN **2** TENNIS	HOLD 10 MIN **3** GAME 3	START 20 MIN **4** GAME 4

Reset is performed via the OSD and the other four console buttons are mapped (via the OSD) to a game controller so ensure you have a controller with plenty of buttons.

The controllers, while not odd at the time given there was little to compare them to, are of an unusual form factor. They each offer 4 directional control like a standard joystick, but could also be pushed in/down, pulled out/up, and rotated to the left and right (not like a paddle/spinner as it would just slightly move [about 10 degrees] in each direction) for 8 different actions (add the four console buttons and that's 12 actions you'll need to map).

The console also sports built in Hockey and Tennis games. Upon turning the console on a friendly G? appears asking you to select among the 4 games by pressing one of the numbered buttons on the console. You can set a time limit by pressing the Time (1) button followed by the button with the time limit you want to set (2/5/10/20 mins) and you can select a mode by pressing the Mode (2) button followed by one of the 4 numbered buttons. Press the 4 button to Start your game.

Game Boy/Game Boy Color [Nintendo] (1989)

The Game Boy was Nintendo's first portable game console and features a monochrome screen and was initially bundled with the smash hit Tetris game cartridge.

Game Boy Advance (2001)

The Game Boy Advance was the next step in the evolution of the Game Boy line of hand held game consoles from Nintendo. It sported the same basic controls as the original with A/B buttons, a DPad and Start/Select buttons.

Intellivision [Mattel] (1979)

The Intellivision was Atari's big competitor in the early days of the Atari 2600 boasting more realistic looking sports games and a controller with much more functionality and a strange disc based joystick substitute. The controller had a 12 button numeric keypad that supported overlays that each game typically included, 4 buttons on its sides and a multi-directional disc that one would press on in place of a more standard joystick (there were add-ons that one might attach on top of this to give a more joystick like feel).

Ensure that you create a directory named 'Intellivision' in the root of your SD cart and include files with a ROM or INT file extension. ZIP files are supported.

Interton VC 4000 (1978)

The Interton Video Computer 4000 is an early German video game console that was released by several different companies under a variety of names throughout Europe. This system is a bit of a predecessor to the Emmerson Arcadia 2001 with both being based on the Signetics 2560 CPU and both having controllers that sported a joystick and 12 button keypad.

Neo Geo Advanced Entertainment System [AES] (1990)

The Neo Geo AES is SNK's home console version of their stand up MVS arcade game system. Later the Neo Geo CD was released with CD-ROM based games instead of ROM cartridges. The controllers were sturdy, arcade style controls with a ball top joystick, Start / Select buttons, and 4 buttons labelled A, B, C, and D for game play. The sheer size of the game cartridges really set this system apart as did its price.

Nintendo Entertainment System / NES [Nintendo] (1983)

The NES was Nintendo's blockbuster game console release following 1983's famous video game crash. The NES core also supports the Famicom Disk System. Its controllers offer Start / Select buttons alongside a D-Pad and two buttons for gaming labelled A and B.

Odyssey II [Magnavox] (1978)

The Odyssey II had self-centering one button analog joystick controllers and the console had a membrane keyboard allowing for some fairly sophisticated (for the later 1970s/early 1980s) strategy games that would come packed with extra parts such as game boards / plastic tokens and were known as the Master Strategy Series. The Odyssey II also had a speech synthesize peripheral called “The Voice” which is supported via this core.

Note that this system didn’t standardize on which joystick was for player 1 so if the default configuration doesn’t work then enable the joystick swapping option in the OSD.

Sega CD / Mega-CD [Sega] (1991)

The Mega CD is actually a CD-ROM add-on for the Sega Genesis and is not a stand-alone console, but had its own CPU. It would attach via an edge card connector on the right side of the Genesis hardware. It brought the world of full motion video (FMV) to Sega Genesis games. Requires CD_BIOS.ROM in the game folder.

Sega Genesis / Mega Drive (1988)

F1 = Reset as Japanese NTSC Console
F2 = Reset as North American NTSC Console
F3 = Reset as European PAL Console

The header on the ROM cartridge images files may be used to detect the region of the cartridge, but you can always force a particular region with extensions of .BIN for Japanese, .GEN for North American, and .MD for European game cartridges.

Sega Master System [SMS] / Game Gear (1985)

The Sega Master System was Sega's answer to Nintendo's NES, but while realizing success in Europe it failed to gain any real traction in the USA. It supported both typical cartridges (top slot) as well as game cards resembling what the TurboGrafx-16 would later use (front to the right of the controller ports). The game controllers were a simple D-Pad with two buttons labelled 1 (Start) and 2. It was released with a light gun just like the NES. Later they released a joystick version of the game pad controller and also a 'sports pad' (a track ball) and eventually 3-D Glasses. The Game Gear is really, more or less, a portable Sega Master System albeit using a different form factor for the game cartridges making them incompatible although there was a converter for playing SMS games on the Game Gear given the hardware similarities.

Sony PlayStation (1994)

Note: This is still in beta with regular build available at https://github.com/MiSTer-unstable-nightlies/PSX_MiSTer/releases/

The Sony PlayStation was a console released with a focus on 3D graphics and is a reasonable mark for the transition to CD-ROM based games (although it was not the first). The console sports a sizable library of games.

The system primarily used a game pad with four buttons the correlated to different colors shapes (vs A, B, X, Y, …), a Start and Select button in the middle and a pair of Left/Right shoulder buttons on the rear of the controller. Later this was upgraded to have two analog thumb sticks. The console also used memory cards to load/save game state.

As above this core is in beta and does not run all game flawlessly although it's an incredibly impressive work as it. Among other titles, Ridge Racer (my first PS game) works great!

Super Nintendo / SNES [Nintendo] (1990)

The Super Nintendo was Nintendo's foray into 16-bit gaming. Its controllers featured Start / Select buttons alongside a D-Pad and four additional game buttons labelled A, B, X and Y. This controller layout should be readily familiar to NES fans given that it just adds two extra buttons.

TurboGrafx-16 / PC Engine [NEC] (1987)

The NEC TurboGrafx-16 was billed as a 16-bit game console and uses flat HuCard style game cartridges that were about the size of a credit card. It also, later, had a CD-ROM add-on. The gamepad controllers featured a D-Pad, Select / Run buttons typically used for starting a game and two additional buttons marked I and II (along with switches above them to turn on rapid fire).

Vectrex [GCE / Milton Bradley] (1982)

The Vectrex was a standalone (i.e. no TV hookup required as the display was part of the console) vector graphics video game console and used color overlays on the screen to add to its graphical flair (pictured with the Minestorm overlay on the screen). Overlay files (.OVR), available on the same site as the main core, must be placed in the same folder as the ROM files for the Vectrex core to display them. The controller consisted of a joystick and four buttons numbered 1-4.

WonderSwan / WonderSwan Color [Bandai] (1999/2000)

The WonderSwan and WonderSwan color are handheld game systems released in Japan from Bandai. Both sport the same basic form factor and control layout. The non-color version features 8 shades of gray.