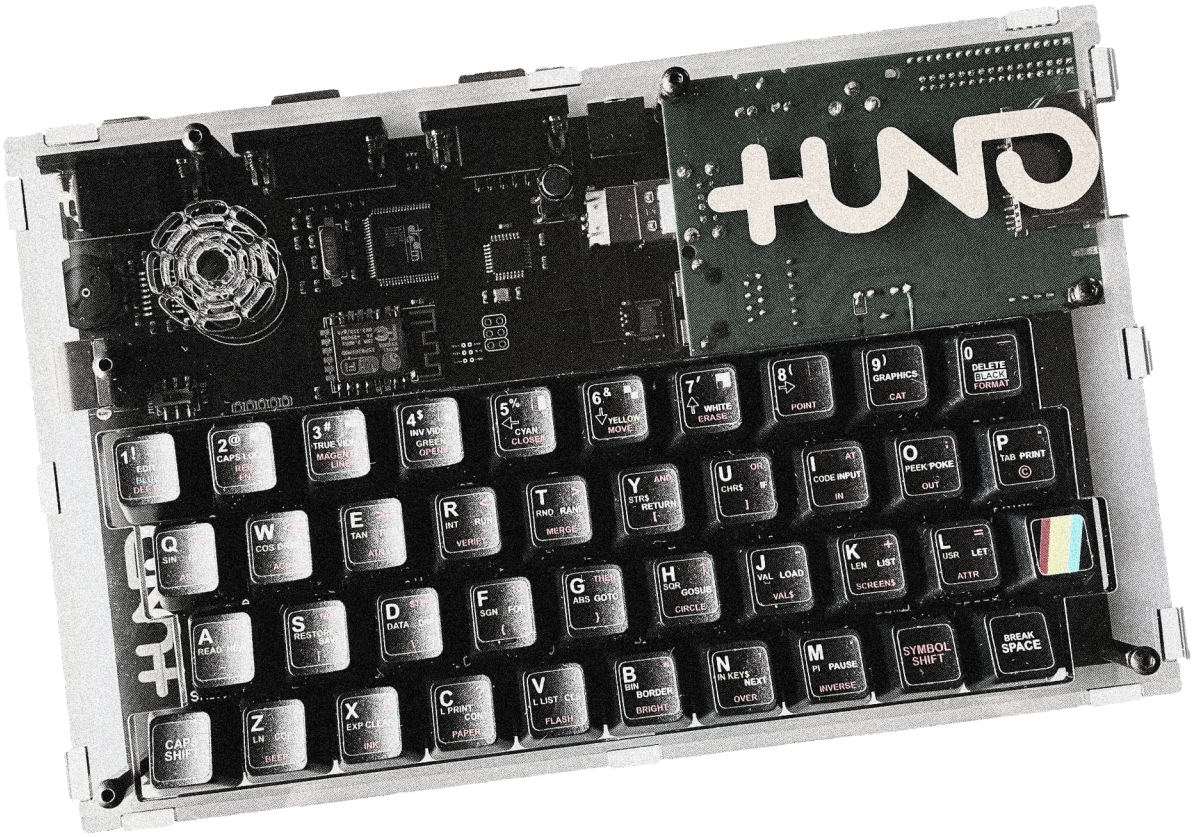
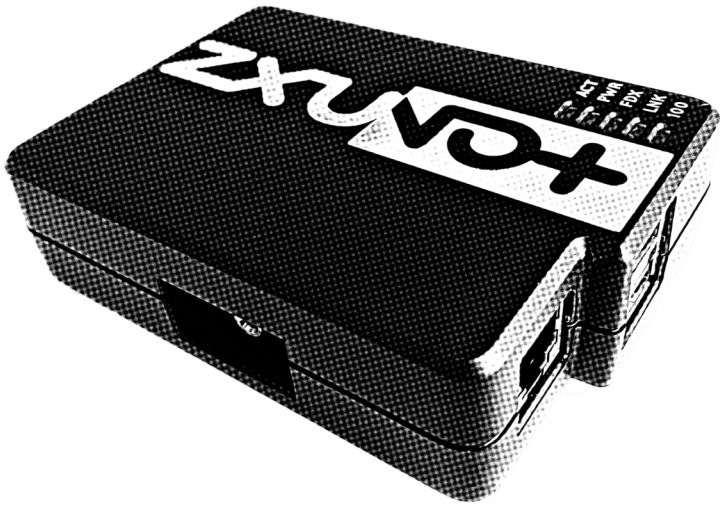
# ZXUNO+

# Manual





# Index

# Introduction

ZXUNO+ and +UNO are another iteration of ZX-Uno a hardware and software project based on an FPGA board programmed to work like a ZX Spectrum computer, and created by the ZX-Uno team: Superfo, AVillena, McLeod, Quest and Hark0.

Over time, the project has been growing, and now it is possible to install different configurations (cores) in the flash memory of the FPGA, which work like different systems than the ZX Spectrum, and you can choose to start the ZXUNO+ with the desired configuration among all those installed.

ZXUNO+ official web page is https://zxuno.speccy.org.

# Acknowledgements

Most of the content is based on information previously shared:

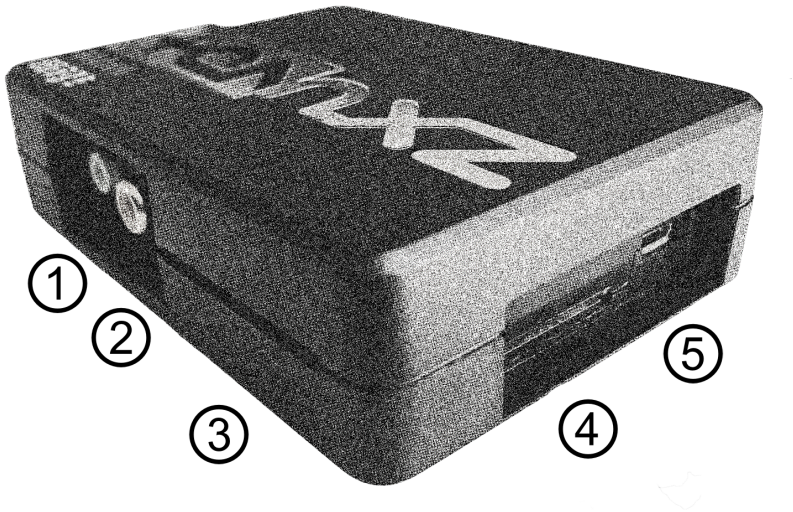
- At ZX-Uno forum
- At foroFPGA
- Several existing FAQ, mostly the original version by @uto_dev, and the latest one by @desUBIKado
- At ZX-Uno Telegram Channel

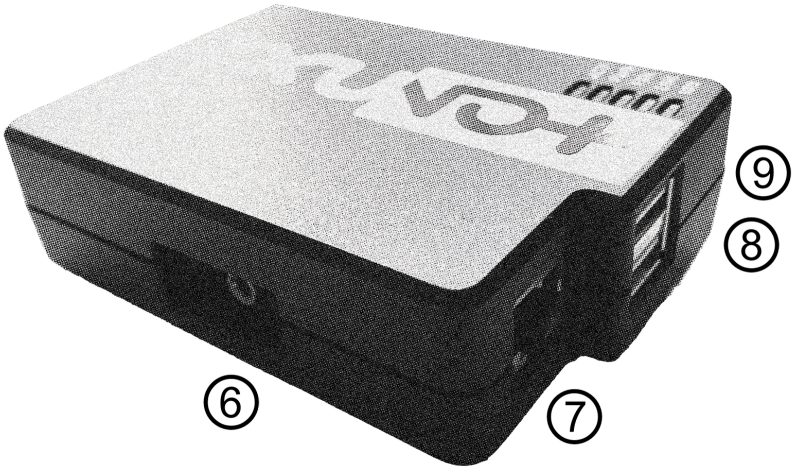
A very special thanks to desUBIKado for the continuous and thorough work finding and sharing information about all the available cores and their functionalities.

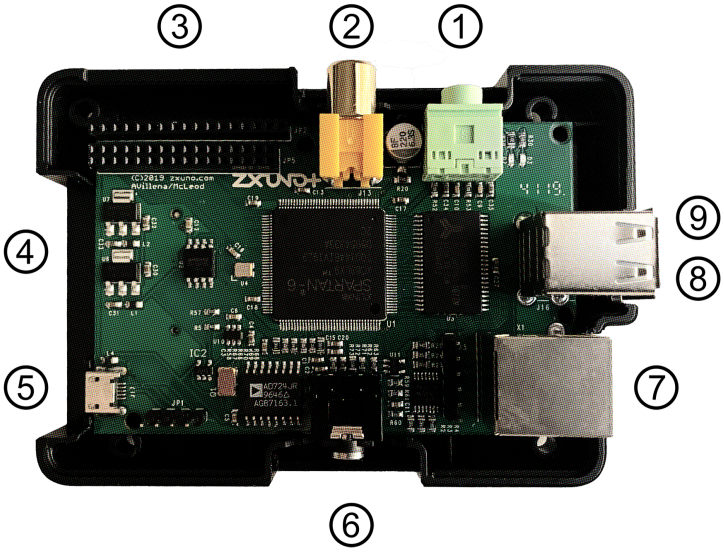
Without the previous work of all of these people (and more), this manual wouldn't exist.

# Ports and Connectors

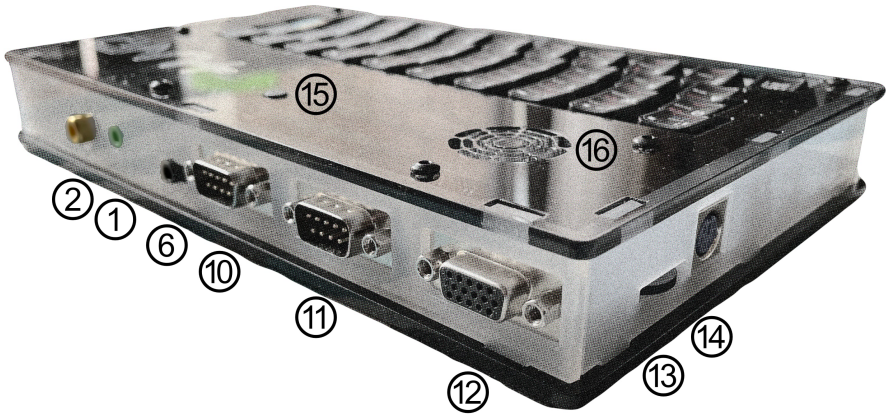
## ZXUNO+



## +UNO

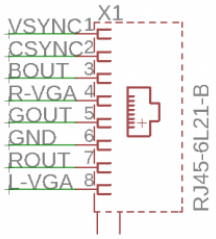




## Description

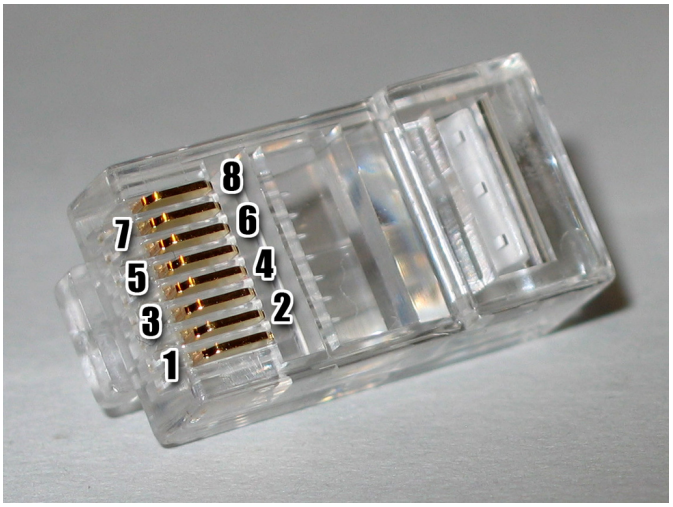
| | |
|---|---|
| 1 | Audio Out |
| 2 | Composite Video Out |
| 3 | Expansion Port |
| 4 | SD Card Slot |
| 5 | Power (USB Mini-A) |
| 6 | Audio In |
| 7 | Video Out (RJ-45) |
| 8 | Mouse USB (PS/2) Port |
| 9 | Keyboard USB (PS/2) Port |
| 10 | Left Joystick Port |
| 11 | Right Joystick Port |
| 12 | RGB/VGA video out |
| 13 | Volume control |
| 14 | PS/2 Port |
| 15 | Reset button |
| 16 | Internal speaker opening |

### RJ-45 Video Out

This is the pinout or RJ-45 connector to use as VGA output on a ZXUNO+:




This are the pin numbers for RJ-45 connector to use as RGB out with a SCART connector on a ZXUNO+:



# Initial Setup

In order to be able to set up and use a ZXUNO+ you need, at least, the following:

- A USB charger or a TV or other device that offers USB power. Usually 500 mA is more than enough
- VGA cable and monitor
- PS/2 keyboard (with USB to PS/2 adapter or connector)

In order to take advantage of its full potential, you may also have:

- A SD card, not necessarily very large
- PC speakers to connect to the audio output, or a stereo jack converter to two red/white RCA connectors to connect to the TV
- A PS/2 mouse (USB to PS/2 adapter needed)
- An audio cable with a stereo 3.5 mm jack on one side, and both audio channels split into two mono outputs on the other side, if you want to use an audio player and/or recorder, like, for example, a Miniduino (see more info later)., a PC/Mac/Raspberry PI, etc. or a cassette tape recorder/player. The right sound channel is used as input (EAR) and the left channel can be used as output (MIC).

## SD card formatting

In order to use a SD card, it has to be formatted with, at least, one FAT16 or FAT32 format (depending on the case, one or the other format is recommended for compatibility with different third-party cores). It must be the first partition if there are more than one, except for the Spectrum core which can have the first partition in +3DOS format, and then the second one in FAT16 or FAT32 format to use with a +3e ROM.

FAT16 partitions have a maximum size of 4GB

When naming a partition which will be used with esxdos, it's important not to use the same of any directory inside, or an access error will happen when trying to see the contents (e.g. do not name the partition as `BIN`, `SYS` or `TMP`).

The Spectrum core can also have the first partition in +3DOS format, and then the second one in FAT16 or FAT32 format to use with a +3e ROM.

This table shows special requirements of the cores that use the SD card.

| Core | FAT 16 | FAT 32 | +3e | Primary Partition Type | Extra Partition | Access | Notes |
|---|---|---|---|---|---|---|---|
| ZX Spectrum EXP | Yes | Yes | Yes | Any | Yes | Full | Uses SPI Flash esxdos |
| ZX Spectrum Kyp 48K | Yes | Yes | No | Any | Yes | Full | Uses embedded esxdos |
| ZX Spectrum Kyp 128K | Yes | Yes | No | Any | Yes | Full | Uses embedded esxdos |
| ZX Spectrum TBBlue | Yes | Yes | No | Any | Yes | Full | Can use esxdos in SD |
| Acorn Atom | Yes | No | No | Any | No | Atom software archive | |
| Acorn Electron | Yes | No | No | Any | No | Only `MMB` file | |
| Amstrad CPC 464 | | | | | | | Doesn't use the SD |
| Amstrad CPC 6128 | No | Yes | No | 0x0b (Win95 FAT-32) | No | Only disk image files `.DSK` in SD card root | 4G or less partition and 4096 cluster |
| Apple I | | | | | | | Doesn't use the SD |
| Apple II | No | No | No | No | Exclusive | No | Exclusive special format |
| Arcades | | | | | | | SD not used |
| Atari 800XL | No | Yes | No | Any | No | Only `ROM`, disk or cartridge images | |
| Atari 2600 | Si | No | No | Any | No | Only `ROM` files | |
| BBC Micro | Yes | Si | No | Any | No | Only `MMFS` or `MMB` file, depending on version | |
| Camputers Lynx | | | | | | | Doesn't use the SD |
| CHIP-8 | Yes | Yes | No | Any | No | Only ROMs (`.BIN` or `.CH8`) | |
| ColecoVision | Yes | No | No | Any | No | Only ROMs (`.ROM`) | |
| Colour Genie | | | | | | | Doesn't use the SD |
| Commodore 16 | No | No | No | No | Exclusive | No | Exclusive special format (C64 compatible) |
| Commodore 64 | No | No | No | No | Exclusive | No | Exclusive special format (C16 compatible) |
| Commodore PET | | | | | | | Doesn't use the SD |
| Commodore VIC-20 | | | | | | | Doesn't use the SD |
| Flappy Bird | | | | | | | Doesn't use the SD |
| Galaksija | Yes | No | No | Any | No | Only `GTP` files | |
| HT-1080Z | Yes | No | No | Any | No | Only`CAS` files | |
| Jupiter ACE | Yes | No | No | Any | No | Only `TAP` files | |
| MSX | Yes | No | No | 0x06 (16-bit FAT) | Yes (FAT16) | Full | Not compatible with PC XT |

| Core | FAT 16 | FAT 32 | +3e | Primary Partition Type | Extra Partition | Access | Notes |
|---|---|---|---|---|---|---|---|
| NES | Yes | Yes | No | Any | No | Only ROMs (.NES) | |
| Ondra SPO 186 | | | | | | | Doesn't use the SD |
| Oric Atmos | No | No | No | No | Exclusive | No | Exclusive special format |
| Oric Atmos (Kyp) | | | | | | | Doesn't use the SD |
| PC XT | Yes | No | No | 0x06 (16-bit FAT) | No | Full. Needs DOS installed | Not compatible with MSX |
| PC XT CGA | Yes | Si | No | Depending on DOS version | No | Full. Needs DOS installed | Not compatible with MSX |
| Pong | | | | | | | Doesn't use the SD |
| SAM Coupé | | | | | | | Doesn't use the SD |
| Sega Master System | Yes | No | No | Any | No | Only SMS or BIN files | |
| SmartROM | Yes | Yes | No | Any | No | Needs specific files | |
| Vectrex | Yes | Yes | No | Any | No | Only BIN or VEC files | |
| Videopac | Yes | No | No | Any | No | Only ROMs (.BIN) | |
| ZX81 | | | | | | | Doesn't use the SD |
| ZX81 y ZX80 | Yes | Yes | No | Any | No | Only image files (.o or .p) | |

### Windows

For simple configurations, and cards of the right size (less than 2GB for FAT16 or less than 32GB for FAT32), you can use the official formatting tool of the SD Association .

For other, more complex, configurations, and depending on operating system version, you may use the command line tool `diskpart` or Windows Disk Management GUI.

For example, on Windows, to format a card with just one FAT16 partition (if the card is 4GB or less in size), which is shown as disk 6 when typing `list disk` on `diskpart`:

```
select disk 6
clean
create part primary
active
format FS=FAT label=ZXUNOPLUS
exit
```

To create two 4GB FAT16 partitions (for example, to use with MSX core) and leave the rest of the space with a third one in FAT32 format (for cards of more than 8GB):

```
select disk 6
clean
create part primary size=4000
set id=06
active
format fs=FAT label=ZXUNOPLUS quick
create part primary size=4000
format fs=FAT label=EXTRA quick
create part primary
format fs=FAT32 label=DATA quick
exit
```

To create one 4GB FAT16 partition (for example, to use with Amstrad CPC 6128 core) and leave the rest of the space with another one in FAT32 format (for cards of more than 4GB):

```
select disk 6
clean
create part primary size=4000
set id=0b
active
format fs=FAT32 label=ZXUNOPLUS unit=4k quick
create part primary
format fs=FAT32 label=EXTRA quick
exit
```

### macOS

For simple configurations, and cards of the right size (less than 2GB for FAT16 or less than 32GB for FAT32), you can use the official formatting tool of the SD Association or Disk Utility, which is included with the operating system.

In other case, you should use the command line.

For example, to format a card, shown as `disk6`, with only one FAT16 partition (if the card size is less than 2GB):

```
diskutil unmountDisk /dev/disk6
diskutil partitionDisk /dev/disk6 MBR "MS-DOS FAT16" ZXUNOPLUS R
```

To split it into two FAT16 partitions of the same size (if the card size is 4GB or less):

```
diskutil unmountDisk /dev/disk6
diskutil partitionDisk /dev/disk6 MBR "MS-DOS FAT16" ZXUNOPLUS 50% "MS-DOS FAT16"
EXTRA 50%
```

To create two FAT 16 partitions (e.g. to use MSX core) and have the rest of space as another FAT32 partition (for cards more than 8GB in size):

```
diskutil unmountDisk /dev/disk6
diskutil partitionDisk /dev/disk6 MBR %DOS_FAT_16% ZXUNOPLUS 4G %DOS_FAT_16% EXTRA 4G
"MS-DOS FAT32" DATA R
sudo newfs_msdos -F 16 -v ZXUNOPLUS -c 128 /dev/rdisk6s1
sudo newfs_msdos -F 16 -v EXTRA -b 4096 -c 128 /dev/rdisk6s2
```

`diskutil` cannot create FAT16 partitions which are bigger than 2G and also format them. That's why, in this example, after only creating the partitions, we have to format them.

To create one FAT32 4GB partition (e.g. to use with Amstrad CPC 6128 core), and then have the rest of space available as a second FAT32 partition (for cards of more than 4GB):

```
diskutil unmountDisk /dev/disk6
diskutil partitionDisk /dev/disk6 MBR "MS-DOS FAT32" ZXUNOPLUS 4G "MS-DOS FAT32" EXTRA
R
```

In this example, since the partition has a size of exactly 4G, macOS will use a cluster size of 4096, which is the one needed for the Amstrad CPC 6128 core. For a smaller size, you may have to format again the first partition with some commands like these:

```
diskutil unmountDisk /dev/disk6
newfs_msdos -F 32 -v ZXUNOPLUS -b 4096 /dev/rdisk6s1
```

The Spotlight feature in macOS you to search the items on the SD card, creating a number of hidden files. You can disable the indexing with these commands (assuming that the SD partition is named `ZXUNOPLUS`):

```
mdutil -i off /Volumes/ZXUNOPLUS
cd /Volumes/ZXUNOPLUS
rm -rf .{,_.}{fseventsd,Spotlight-V*,Trashes}
mkdir .fseventsd
touch .fseventsd/no_log .metadata_never_index .Trashes
cd -
```

### Linux

There are a lot of tools for Linux that can format and/or partition an SD card (`fdisk`, `parted`, `cfdisk`, `sfdisk` or `GParted` to name a few). It should only be taken into account that the partition scheme must always be MBR, and the first partition (the one that will be used for esxdos) must be primary partition.

For example, to format a card, shown as `sdc`, with only one FAT16 partition (if the card size is less than 4GB)

```
sudo fdisk --compatibility=dos /dev/sdc
```

```
(...)
Command (m for help): n
Partition type
   p primary (0 primary, 0 extended, 4 free)
   e extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1): 1
First sector (62-31116288, default 62):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (128-31116288, default 31116288):
Created a new partition 1 of type 'Linux'

Command (m for help): t
Selected partition 1
Hex code (type L to list all codes): 6
Changed type of partition 'Linux' to 'FAT16'.

Command (m for help): a
Partition number (1, default 1): 1
The bootable flag on partition 1 is enabled now.

Command (m for help): p
Disk /dev/sdc
Disklabel type: dos
Disk identifier

Device     Boot   Start     End  Sectors   Size Id Type
/dev/sdc1           62 31116288  31116288 984,9M 6 FAT16
```

Format the FAT partition (requires root privileges)

```
sudo mkfs.fat -F 16 -n ZXUNOPLUS -s 128 /dev/sdc1
```

# Keyboard and mouse

## +UNO keyboard

The +UNO keyboard, being similar to the original ZX Spectrum keyboard, lacks some of the existing keys on a modern PC keyboard. The keyboard membrane is connected to an Arduino board, which manages the transformation key presses to PS/2 keyboard protocol. The board is programmed so it can behave in different modes according to your needs.

The default is ZX Spectrum mode. To change to a different mode, you must press `Caps Shift+Symbol Shift+U` and then the key for the desired mode. After doing that, some text is automatically typed, to show the selected mode (for example `.zx` if you press `Caps Shift+Symbol Shift+U` and then `0`).

This table shows the available modes and activation keys:

| Mode | Key |
|---|---|
| ZX Spectrum | `0` |
| Amstrad CPC | `1` |
| MSX | `2` |
| Commodore 64 | `3` |
| Atari 800XL | `4` |
| BBC Micro | `5` |
| Acorn Electron | `6` |
| Apple (I and II) | `7` |
| Commodore VIC 20 | `8` |
| PC XT | `9` |
| Oric Atmos | `A` |
| SAM Coupé | `B` |
| Jupiter ACE | `C` |

ZX Spectrum mode key assignment, with the corresponding keypress when used simultaneously with `Caps Shift+Symbol Shift`:

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| F1 | F2 | F3 | F4 | F5 | F6 | F7 | F8 | F9 | F1 |
| **Q** | **W** | **E** | **R** | **T** | **Y** | **U** | **I** | **O** | **P** |
| F11 | F12 | | | | | Mode | | | |
| **A** | **S** | **D** | **F** | **G** | **H** | **J** | **K** | **L** | **Enter** |
| | | | | ScrLk | | | | | |
| **CShift** | **Z** | **X** | **C** | **V** | **B** | **N** | **M** | **SShift** | **Space** |
| | | Save | | Vers | hRes | sRes | | | |

Where:

- `ScrLk`: `Scroll Lock` changes between RGB and VGA video mode (on Next Core, you must use `Caps Shift+Symbol Shift+2` or `F2` instead)
- `Save`: Sets the current mode as the default one
- `Vers`: Shows (types) current firmware version
- `hRes`: Hard Reset
- `sRes`: Soft Reset

The full list of key combinations (and compatible modes) is as follows:

| Caps S.+Symbol S. | Mode | Action |
|---|---|---|
| 1 | All | F1 |
| 2 | All | F2 |
| 3 | All | F3 |
| 4 | All | F4 |
| 5 | All | F5 |
| 6 | All | F6 |
| 7 | All | F7 |
| 8 | All | F8 |
| 9 | All | F9 |
| 0 | All | F10 |
| Q | All | F11 |
| W | All | F12 |
| S | C64 | Ctrl+F12 |
| E | Acorn/CPC | PgUp |
| R | Acorn | PgDown |
| U | All | Mode |
| G | ZX/MSX/C64 | ScrLk |
| X | All | Save |
| C | PC | OPQA |
| V | All | Version |
| B | ZX | Ctrl+Alt+Bcksp |
| N | ZX | Ctrl+Alt+Supr |

## PS/2 Keyboard

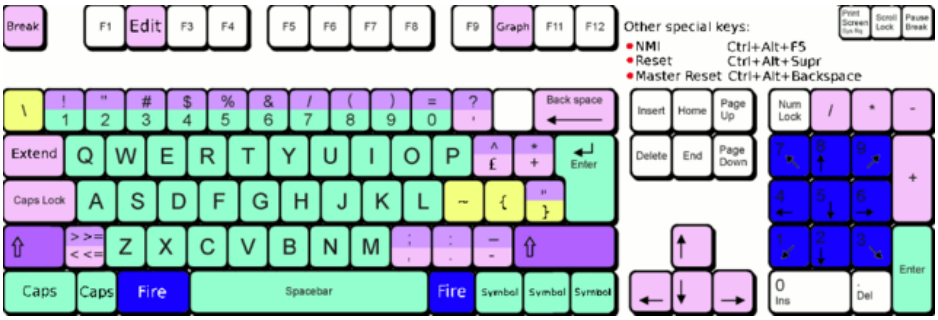
The keyboard map (physical keys of the keyboard assignment to the keystrokes that are presented to the different cores) is changed using the `Advanced` menu of the BIOS. There are three different maps to choose from: Spanish (default), English, and Spectrum (advanced).

You can also change it using the `keymap` utility. Inside `/sys` you have to create a directory named `keymaps` and copy inside the keyboard map files that you want to use. For example, to switch to the US map you have to write `.keymap us` from esxdos.

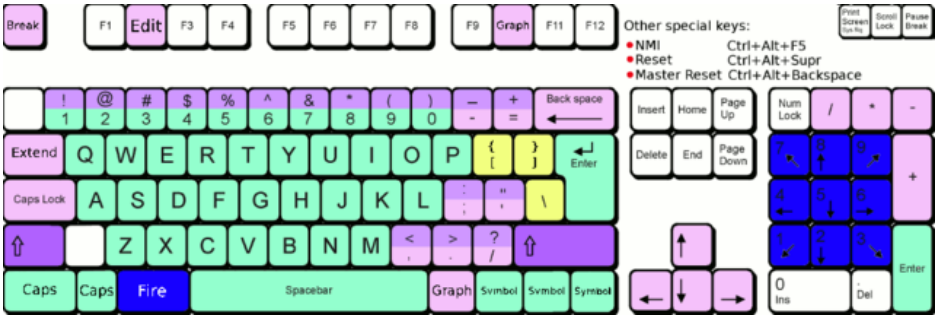
For the map to be preserved after a master reset, it has to be selected as `Default` in the BIOS.

For more information, see this message in the ZX-Uno forum.

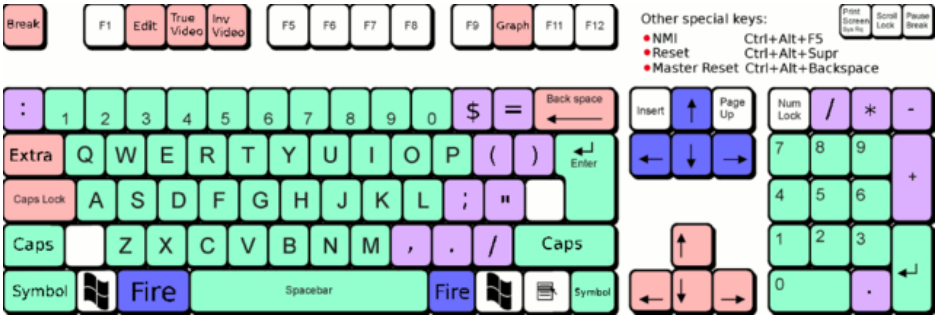
### Spanish




### English




### Spectrum



## Special keys and buttons

Special keys which can be used during startup:

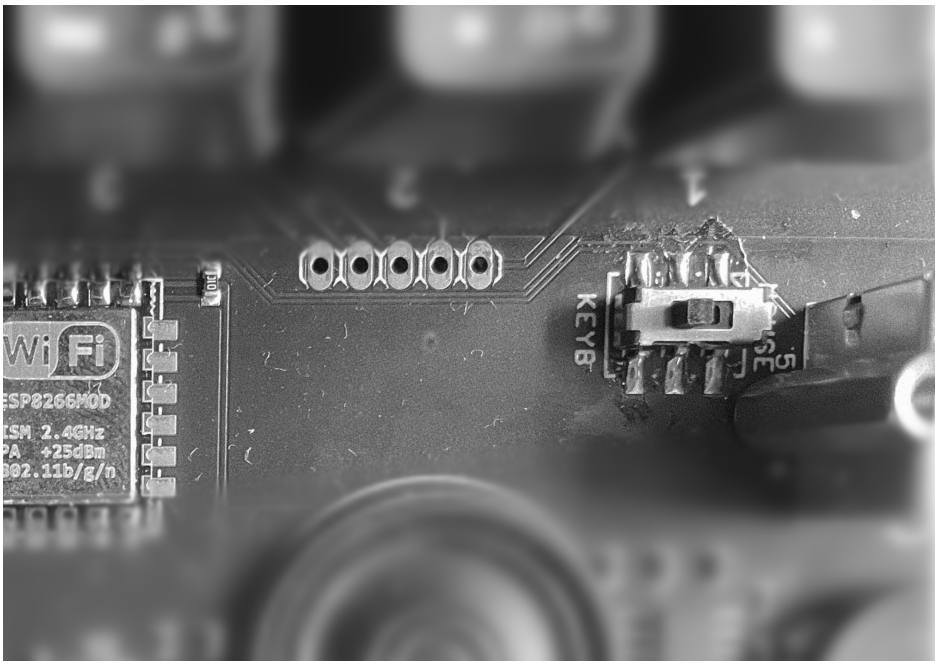
- F2 (Caps Shift+Symbol Shift+2 on +UNO): Enter BIOS setup
- Caps Lock or Cursor down or, if a joystick is connected, pressing down: Core selection menu
- Esc, or if a joystick with two or more fire buttons is connected, pressing the 2nd fire button: ZX Spectrum core ROM selection menu
- R: Loads the Spectrum core ROM in "real" mode, disabling esxdos, new graphics modes, etc. (pressing Esc afterwards allows you to choose another core)
- / (numeric keyboard): Load the default ZX Spectrum core ROM in "root" mode (pressing Esc afterwards allows you to choose another core)
- Number from 1 to 9: Load the core in the flash location corresponding to that number

Special keys that can be used while running the main core (ZX Spectrum):

- Esc: BREAK
- F2 (Caps Shift+Symbol Shift+2 on +UNO): Edit
- F3 (Caps Shift+Symbol Shift+3 on +UNO): True Video
- F4 (Caps Shift+Symbol Shift+4 on +UNO): Inverse Video
- F5 (Caps Shift+Symbol Shift+5 on +UNO): NMI
- F7 (Caps Shift+Symbol Shift+7 on +UNO): Play or pause when playing .PZX files
- F8 (Caps Shift+Symbol Shift+8 on +UNO): Rewind .PZX file to the previous mark
- F10 (Caps Shift+Symbol Shift+0 on +UNO): Graph
- F12 (Caps Shift+Symbol Shift+W on +UNO): Turbo Boost. Speeds up CPU to 28MHz while pressed (beginning with core EXP27).
- Ctrl+Alt+Backspace (Caps Shift+Symbol Shift+B on +UNO): Hard reset. Backspace is the delete key, located in the top-right portion of the keyboard, above Enter.
- Ctrl+Alt+Del (Caps Shift+Symbol Shift+N on +UNO): Soft reset.
- Scroll Lock: Switches between composite and VGA video modes.

## PS/2 Mouse

You can also connect a mouse to the PS/2 port. The switch inside a +UNO, near the Wi-Fi chip and keys 1 and 2, selects between a direct connection of the mouse, or if a PS/2 splitter cable is needed (when using the mouse along with a PS/2 keyboard).



# esxdos

esxdos is a firmware for the DivIDE/DivMMC hardware interfaces (which ZXUNO+ implements). This allows access to storage devices such as a SD card. It includes commands similar to those of UNIX, although to use them you must precede them with a period, for example `.ls`,` .cd `, `.mv`, etc.

For it to work, it is necessary to include the corresponding files in the first partition of the SD card.

At the time of writing this document, the version included with ZXUNO+ is 0.8.6, and it can be downloaded from the official website at this link.

Once downloaded and extracted, you have to copy the directories `BIN`, `SYS` and `TMP`, and all of their content, to the root of first partition of the SD card.

If everything has been done correctly, when you turn on the ZXUNO+ Spectrum core, you will see how esxdos detects the card and loads the necessary components to work.

It is also recommended to add the specific esxdos commands for ZXUNO+. These can be obtained from the project source page (here, here and here), and are as follows:

```
back16m
corebios
dmaplayw
esprst
iwconfig
joyconf
keymap
loadpzx
playmid
playrmov
romsback
romsupgr
upgr16m
zxuc
zxunocfg
```

It is explained later what each of them does.

# BIOS


Pressing the F2 key during boot will access the BIOS setup. The BIOS firmware is the first program that runs when the ZXUNO+ is turned on. The main purpose of BIOS is to start and test the hardware and load one of the installed cores.

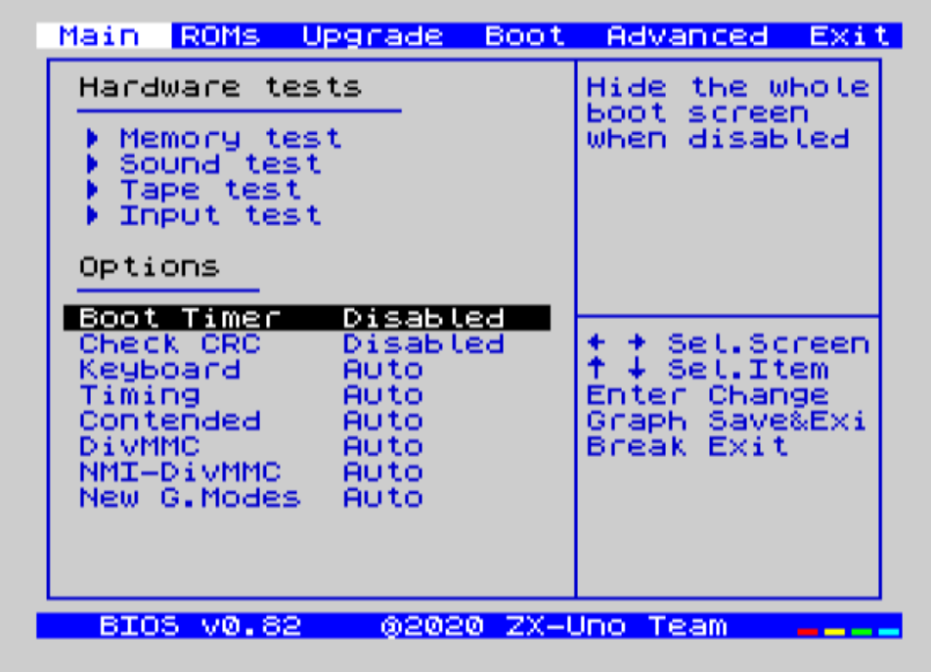
Using left and right cursor keys, you can navigate through the BIOS setup screens. With up and down keys you can choose the different elements of each screen and, with the Enter key, it is possible to activate and choose the options for each of these. The Esc key is used to close open option windows without applying any action.

You can also access the BIOS if you have an adapter and a joystick connected, pressing the down direction button on startup. This accesses the list of installed cores. Choose the latest option (Enter Setup) to access.

Other useful keys to use on startup are:

- Scroll Lock or Cursor Down, or, if a joystick is connected, pressing the down direction button: Core Selection Menu
- Esc or, if a joystick with two or more buttons is connected, pressing the second fire: ZX Spectrum Core ROM selection menu
- R: Load the ZX Spectrum default ROM in "real" mode disabling esxdos, new graphic modes, etc. (combined with Esc let's you use a different ROM)
- / (numeric keypad): Load the ZX Spectrum default ROM in in "root" mode (combined with Esc let's you use a different ROM)
- Number between 1 and 9: Load the corresponding core from Flash

## Main

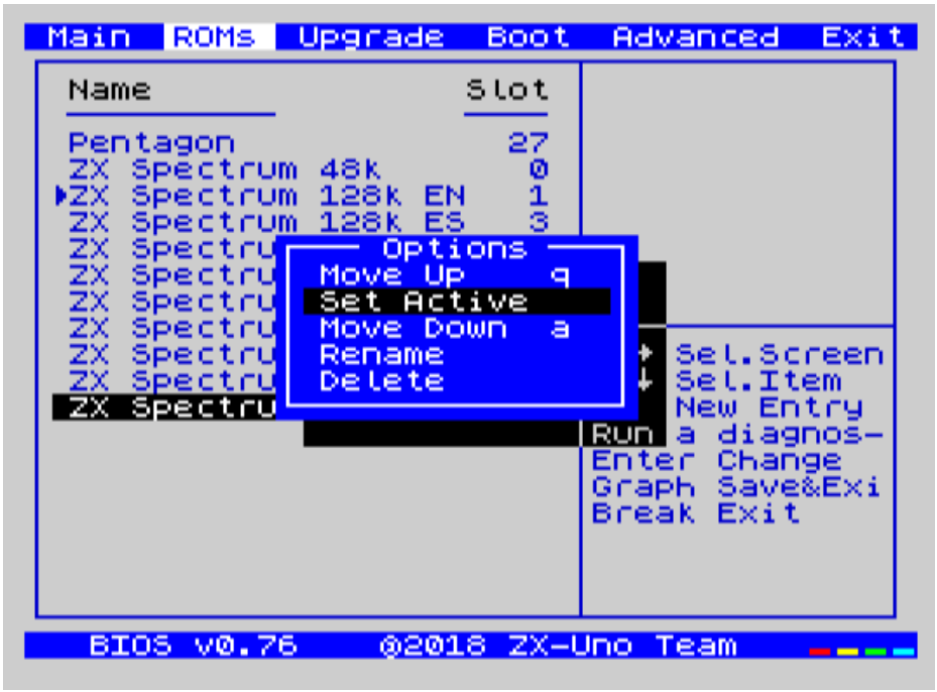



In the first configuration screen, in addition to being able to run several tests, you can define the default behavior for the following:

- Boot Timer: Sets how long the boot screen is available (or hiding it completely)
- Check CRC: Check ROM integrity when loading (more secure) or bypassing it (faster)
- Keyboard
- Timing: ULA Behaviour (48K, 128K, Pentagon Modes)
- Contended
- DivMMC
- DivMMC NMI Support
- New Graphic Modes Support (ULAPlus, Timex, Radastan)

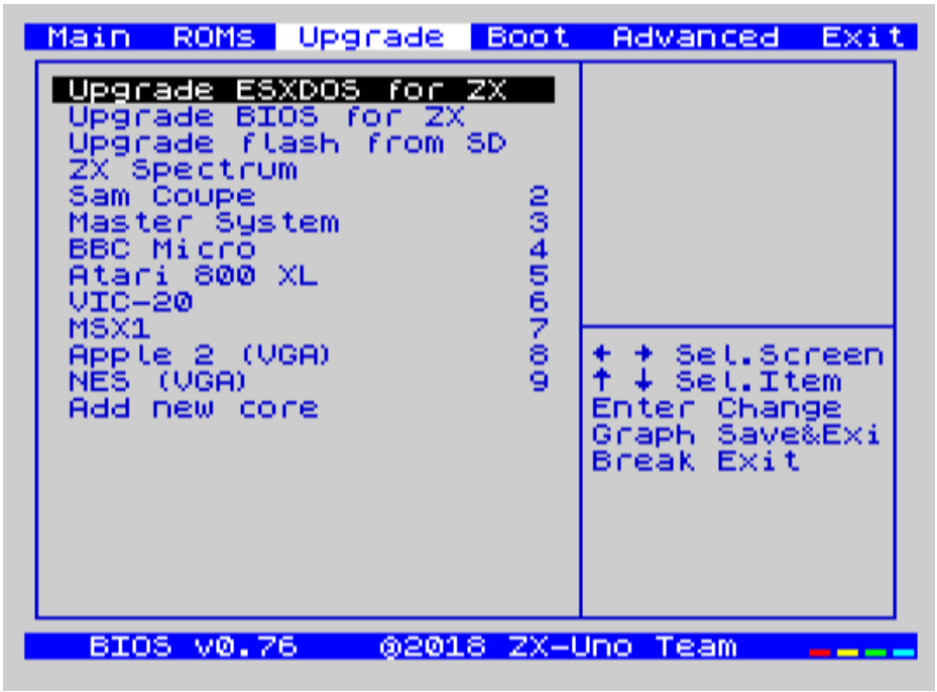
More technical information can be found on the ZX-Uno Wiki.

## ROMs



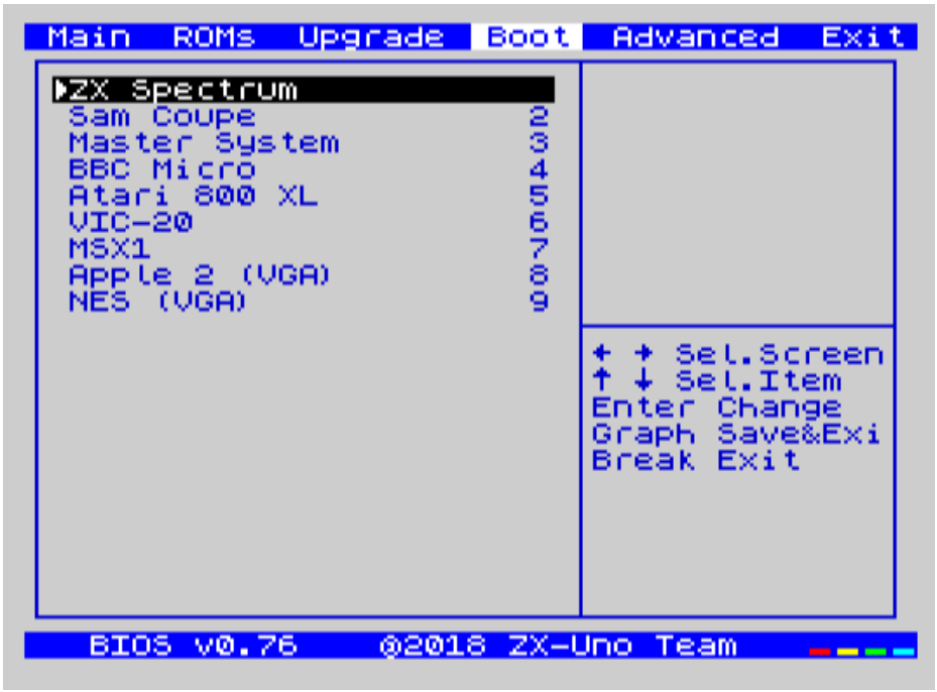

The second screen shows the installed ZX Spectrum ROMs. You can reorder (Move Up, Move Down), rename or delete each of them, as well as choose the one that will be loaded by default at startup (Set Active ).

## Upgrade




*Upgrade* screen is used to perform the different updates of the Flash memory content: esxdos, BIOS, Cores, etc. (see the section corresponding to upgrades for more information).

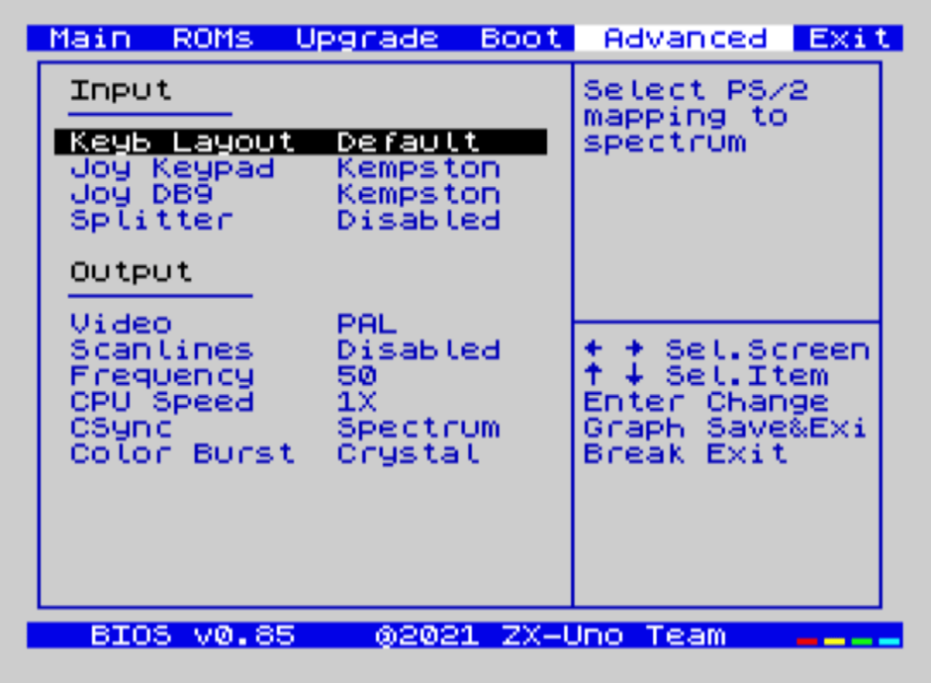
## Boot




In the *Boot* screen you can choose which one of the installed cores is loaded by default at startup.

Never set a Spectrum core (apart from the main one, the first core in the flash. This would set the ZX-Uno into a loop of core resets.

## Advanced



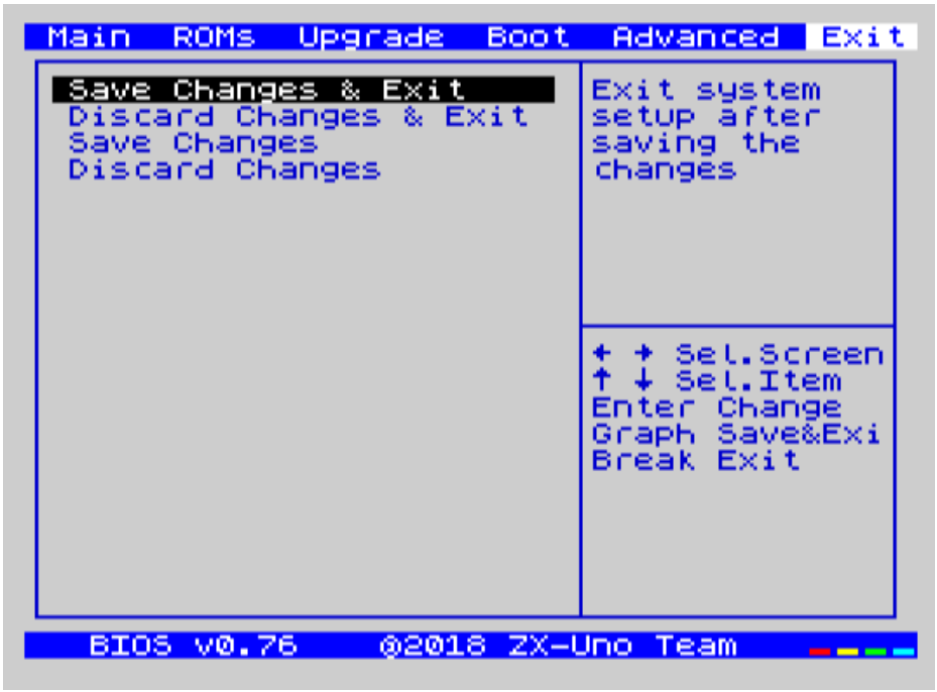

The Advanced configuration screen is used to edit the following settings:

- Keyboard layout (Keyb Layout): See the corresponding section for more information)
- Joystick behavior when emulated with the numeric keypad (Joy Keypad) and the second joystick (with splitter): Kempston, Sinclair Joystick 1, Sinclair Joystick 2, Protek or Fuller
- Behavior of a joystick connected to the main port (Joy DB9): Kempston, Sinclair Joystick 1, Sinclair Joystick 2, Protek, Fuller or simulate the keys Q,` A`, O,` P`, Space and M
- Enable the second physical joystick (splitter). Only on +UNO and for cores which have splitter support
- Video output: PAL, NTSC or VGA
- Scanlines simulation: Enabled or Disabled
- VGA horizontal frequency: 50, 51, etc.
- CPU speed: Normal (1x) or accelerated (2X, 3X, etc.)
- Csync: Spectrum or PAL

This table explains how the joystick buttons are assigned:

| Button | Kempston | Sinclair 1 | Sinclair 2 | Protek | Fuller | QAOPSpcM |
|---|---|---|---|---|---|---|
| Left | Bit 0 | 1 | 6 | 5 | Bit 2 | O |
| Right | Bit 1 | 2 | 7 | 8 | Bit 3 | P |
| Down | Bit 2 | 3 | 8 | 6 | Bit 1 | A |
| Up | Bit 3 | 4 | 9 | 7 | Bit 0 | Q |
| Fire 1 | Bit 4 | 5 | 0 | 0 | Bit 4 | Space |
| Fire 2 | Bit 5 | X | Z | 9 | Bit 5 | M |

## Exit




Finally, from the last screen you can:

- Exit BIOS configuration saving changes (in some cases you will also need a power reset)
- Discard changes and exit
- Save changes without exiting
- Discard Changes

# ZX Spectrum

The main core is the one implementing a ZX Spectrum computer. This core is special, and it cannot be replaced for another that is not a ZX Spectrum, since the ZXUNO+ uses it for its operation.

These are some of its main characteristics:

- ZX Spectrum 48K, 128K, Pentagon and Chloe 280SE implementation
- ULA with ULAplus, Timex and Radastan modes (including hardware scroll and selectable palette group)
- Ability to disable memory contention (for Pentagon 128 compatibility)
- Ability to choose the keyboard behavior (issue 2 or issue 3)
- Possibility to choose the timing of the ULA (48K, 128K or Pentagon)
- Control of screen framing, configurable for type of timing, and possibility to choose between original Spectrum synchronism or progressive PAL standard.
- Timex horizontal MMU support with HOME, DOC and EXT banks in RAM.
- Programmable raster interruption in line number, for any TV line.
- Possibility of activating/deactivating memory bank management registers, for better compatibility with each implemented model
- Ability to activate / deactivate the devices incorporated into the core to improve compatibility with certain programs
- ZXMMC and DIVMMC support for + 3e, esxdos and compatible firmwares
- Turbo Sound support
- SpecDrum support
- Covox support
- Each channel A, B, C of the two AY-3-8912, beeper and SpecDrum chips can be directed to the left, right, both or neither outputs, allowing the implementation of configurations such as ACB, ABC, etc.
- Real joystick and keyboard joystick support with Kempston, Sinclair 1 and 2, Cursor, Fuller and QAOPSpcM protocol.
- Turbo mode support at 7MHz, 14MHz, 28MHz
- Keyboard support (PS/2 protocol) and user-configurable mapping from within Spectrum itself.
- PS/2 mouse support emulating the Kempston Mouse protocol.
- Possibility of video output in composite video mode, RGB 15kHz (PAL and NTSC), or VGA.
- User selectable vertical refresh rate to improve compatibility with VGA monitors.
- Multicore boot support: from the Spectrum you can select an address of the SPI Flash and the FPGA will load a core from there.
- Loading `PZX` tape files from SD

# ROMs

The ZX Spectrum core can be initialized using different ROM versions (48K, 128K, Plus 2, etc.). These are stored in the flash memory of the ZXUNO+, and you can choose which one to load by pressing the Esc key during boot. You can also define the ROM that you want to load by default using the BIOS setup.

See the updates section for more information on how to expand or modify the ROMs stored in flash memory.

## DerbyPro

DerbyPro or Derby++ is an enhanced firmware ROM for the ZX Spectrum based on v1.4 of the Derby development ROM. The Spectrum 128 (codename "Derby") was a Spanish machine commissioned by Investronica and launched in 1985. It came with a keypad that provided additional editing keys. In 1986 the UK version came out with a simplified version of 128 BASIC and no keypad. Derby++ is developed from the Spanish ROM to include the benefits of both versions without the drawbacks and support for new hardware developments.

Features include:

- 100% binary compatible 48K mode.
- 6-channel PLAY commmand.
- Access the esxDOS NMI browser from the boot menu.
- Debugged 128 BASIC with additinoal commands and full scren string editor.
- esxDOS support in 128 BASIC.
- Menu access to TR-DOS.
- PALETTE command for ULAplus.
- Run most Spectrum software without the need to switch configuration in the BIOS.

You can download the ROM, a user manual and other files from the official Facebook Public Group.

Because it's a 64K ROM with support for new hardware these flags can be used when adding it to the SPI flash:

| Flag | Meaning |
|---|---|
| `d` | Enable divMMC |
| `n` | Enable NMI divMMC (esxDOS Menu) |
| `t` | Use 128K timings |

## CargandoLeches

CargandoLeches is a set of ZX Spectrum ROMs that started as a project to load games in any Spectrum model 15-20x faster. No tape is needed, but a digital audio source, as a computer, mobile device, MP3 player, etc. The new ROM detects the loading method and reverts to the original ROM code if needed. This is handled transparently, with no user or program intervention.

Since version 2.0 the project changed from a single ROM to more, each one with different options. This way, you can choose a different mix of options that may include:

- Ultrafast loading
- Reset & Play (After a software reset of the core, the system is ready to load from tape)
- POKE editor
- Enable or disable Sinclair BASIC token expansion

The whole ROM set is available to download from the repository in GitHub here.

Depending on which ROM you choose, the flags when adding to the SPI flash may vary. For example, for the ROM `48le_ea_re_po` (with all features enabled), these flags can be used (we cannot enable NMI DivMMC since the POKE editor will use it):

| Flag | Meaning |
|---|---|
| `d` | Enable DivMMC |
| `h` | Disable ROM high bit (1FFD bit 2) |
| `l` | Disable ROM low bit (7FFD bit 4) |
| `x` | Disable Timex mode |

#### POKEs

When using a ROM with POKE option enabled:

1. Once the game is loaded, after pressing NMI a field will appear in the upper left corner of the screen
2. Enter the POKE address and press `Enter`
3. Enter the POKE value and press `Enter` again
4. Repeat steps 2. and 3. until all the desired POKEs are entered. To finish and return to the game, press `Enter` twice

### Preparing ultrafast loading tapes

The ROMs with ultrafast loading enabled, need special tape audio data which is made from normal loading TAP files, without protections or turbo loading.

In order to create an ultrafast loading tape you need leches and CgLeches command line utilities. Those can be obtained, for Windows, from the official repository. You can also obtain an unofficial version for macOS from this other repository.

In any other case, you can compile from the source code at the official repository. For example, in Linux, to compile using gcc you only need these commands:

```
gcc leches.c -o leches
gcc CgLeches.c -o CgLeches
```

To create an ultrafast loading tape you have to use the CgLeches command from a terminal, giving, at least, the path to the original TAP file and also to the new file to create (WAV or TZX). There are also some other optional parameters, like the loading speed, between 0 and 7 (where 0 is fastest but also more incompatible), if you want to create a mono or stereo file (when making a WAV), and more.

Thus, to make a WAV file with an ultrafast loading tape from the file Valley.tap, with loading speed 5, you could type:

```
(...) CgLeches Valley.tap Valley.wav 5
```

This way, the file Valley.wav can be played from a computer or another device and load using the ROM (see the section about loading from tape for more info).

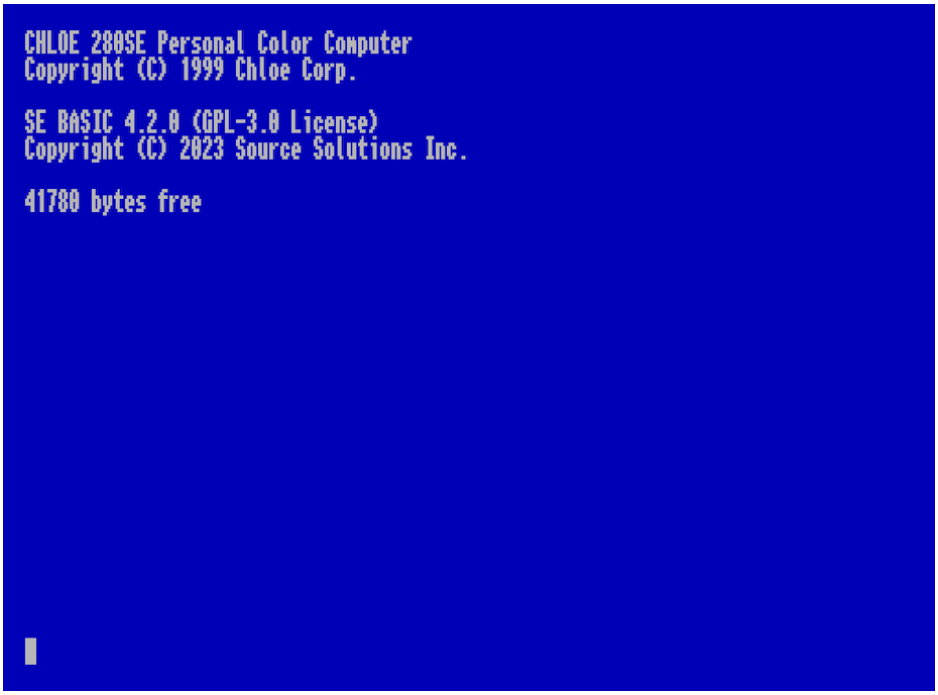
Due to hardware limitations, TZX files made with CgLeches do not work with a Miniduino, although they usually work with PlayTZX.

## SE Basic IV

SE Basic IV is a free open-source BASIC interpreter for the Z80 architecture. Although it aims for a high degree of compatibility with Microsoft BASIC, there are some differences. It's designed to run on the Chloe 280SE but it's also compatible with the ZX Spectrum core of the ZXUNO+.

SE BASIC began development in 1999 as the firmware for the ZX Spectrum SE, the ancestor of the Chloe 280SE. Early versions were patches applied to the original Spectrum ROM. From version 1, it used its own assembly file. From version 2, it added support for ULAplus.

Version 3 (OpenSE BASIC) replaced the original ROM code with an open source version derived from the ZX81 and SAM Coupé ROMs. It's still maintained as an open source replacement firmware for the Spectrum, and is included in the main Debian repository for use with emulators.




Version 4.0 added support for 80 column mode. Version 4.1 was an unsuccessful attempt to refactor the code. Starting in 2019, the latest version (4.2 Cordelia) was rebuilt from the ground up to take full advantage of the ZX Spectrum core of the ZX-Uno (and ZXUNO+). While earlier versions retained a high level of compatibility with Sinclair BASIC and software, this version has no support for Sinclair software and is closer in dialect to Atari BASIC.

Version 4.2 requires that divMMC support is enabled with esxDOS or UnoDOS 3 installed. However, "dot" command commands and the NMI browser are not supported.

Features include:

- 40 column (16 colour) and 80 column (2 colour) paletted video modes.
- Always-on expression evaluation (use variables as filenames).
- Application package format with support for turning BASIC programs into apps.
- Automatic data typing.
- Bitwise logic (AND, NOT, OR, XOR).
- Built-in help system.
- Choice of Microsoft (LEFT$, MID$, RIGHT$) or Sinclair (TO) string slicing.
- Composable characters (supports Vietnamese).
- Disk-based filesystem (no tapes).
- Error handling (ON ERROR…, TRACE).
- Flow control (IF…THEN…ELSE, WHILE…WEND).
- Full random file access from BASIC (OPEN, CLOSE, SEEK).
- Full-size keyboard support (DEL, HOME, END and so on).
- Graphics commands in 40 column mode (CIRCLE, DRAW, PLOT).
- Localisation of character sets, error messages, and keyboard layouts.
- Long variable names.
- Motorola style number entry (%; binary, @; octal, $; hexadecimal).
- NMI BREAK.
- On-entry syntax checking.
- PLAY command with 6-channel PSG and MIDI support.
- Recursive user-defined functions.
- Smart firmware updates.
- Token abbreviation and shortcuts (&; AND, ~; NOT; |; OR, ?; PRINT, '; REM').
- Undo NEW (OLD).
- User-defined channels.
- User-defined character sets (256 characters).
- User-defined macros.
- User-defined screen modes.

For the smart firmware update option to work, SE Basic IV must be installed in the second and third 16K ROM slots.

Using the smart firmware update feature replaces the version of esxDOS you're using with the latest version of UnoDOS 3.

## Other ROMs

Here are flag settings which work when adding to the SPI flash some other known custom ROMs:

| ROM Name | Flags |
|---|---|
| Gosh Wonderful ROM v1.33 | dnhl17x |
| Looking Glass 1.07 | dnhl17x |
| ZX82 by Daniel A. Nagy | dnhl17 |
| ZX85 by Daniel A. Nagy | dntmh1 |
| Arcade Game Designer 0.1 | thl17x |

# SD advanced format (+3e)

ZX Spectrum +3e is one ROM that can be used with ZX Spectrum core. This is an improved Sinclair ZX Spectrum +3, which can use hard disks or memory cards.

+3e uses its own partition format (called IDEDOS), to split de hard disk into several partitions to store data. ROM version 1.28 and later can share IDEDOS partitions with MBR partitions. In other case, you must reserve the whole card for IDEDOS partitions.

The following partition scheme can only be used with ZX Spectrum core.

Each partition in IDEDOS can be between 1 and 16 Megabytes (16 million bytes) in size, and each disk can have between 1 and 65535 partitions. This means that the maximum space used in a card is about 1 TB.

This is one method to split a card into two or three parts, with the first partition IDEDOS (1GB), the second one FAT16 (4GB) and the third one FAT32 (using the remaining space in the card).

exsdos and other programs can be installed into the second partition as explained earlier.

## Windows

You can use Windows Disk Management utility. The steps are:

1. Remove all partitions from the card
2. Create a new extended partition, using the desired space for IDEDOS
3. Create a primary partition, 4GB in size, and format as FAT16
4. Optionally, create another primary partition using the remaining space and format as FAT32

## macOS

You will have to use the command line. The first task is to find out which device is the disk to format:

```
diskutil list
```

For this example, it will be disk 6:

```
(...)
/dev/disk6 (external, physical):
 #: TYPE NAME SIZE IDENTIFIER
 0: FDisk_partition_scheme *15.9 GB disk6
 1: DOS_FAT_32 UNKNOWN 15.9 GB disk6s1
```

Instruction steps:

1. Unmount the disk and edit the partition scheme (the second step requires admin privileges):

```
diskutil unmountDisk /dev/disk6
sudo fdisk -e /dev/rdisk6
```

```
fdisk: could not open MBR file /usr/standalone/i386/boot0: No such file or directory
Enter 'help' for information
fdisk: 1> erase
fdisk:*1> edit 1
Partition id ('0' to disable)  [0 - FF]: [0] (? for help) 7F
Do you wish to edit in CHS mode? [n]
Partition offset [0 - 31116288]: [63] 128
Partition size [1 - 31116287]: [31116287] 2017152

fdisk:*1> edit 2
Partition id ('0' to disable)  [0 - FF]: [0] (? for help) 06
Do you wish to edit in CHS mode? [n]
Partition offset [0 - 31116288]: [2017280]
Partition size [1 - 29099135]: [29099135] 7812504

fdisk:*1> flag 2

fdisk:*1> edit 3
Partition id ('0' to disable)  [0 - FF]: [0] (? for help) 0B
Do you wish to edit in CHS mode? [n]
Partition offset [0 - 31116288]: [9829784]
Partition size [1 - 21286504]: [21286504]

fdisk:*1> print
         Starting       Ending
 #: id  cyl  hd sec -  cyl  hd sec [     start -       size]
------------------------------------------------------------------------
 1: 7F 1023 254  63 - 1023 254  63 [       128 -    2017152] <Unknown ID>
 2: 06 1023 254  63 - 1023 254  63 [   2017280 -    7812504] DOS > 32MB
 3: 0B 1023 254  63 - 1023 254  63 [   9829784 -   21286504] Win95 FAT-32
 4: 00    0   0   0 -    0   0   0 [         0 -          0] unused

fdisk:*1> write
fdisk: 1> quit
```

2. Format the FAT partitions (admin privileges required)

```
diskutil unmountDisk /dev/disk6
sudo newfs_msdos -F 16 -v ZXUNOPLUS -c 128 /dev/rdisk6s2
sudo newfs_msdos -F 32 -v EXTRA -c 128 /dev/rdisk6s3
```

3. Confirm that the new partition scheme has been applied:

```
diskutil list
```

```
(...)
/dev/disk6 (external, physical):
   #:                       TYPE NAME                    SIZE       IDENTIFIER
   0:     FDisk_partition_scheme                        *15.9 GB    disk6
   1:                       0x7F                         1.0 GB     disk6s1
   2:                 DOS_FAT_16 ZXUNOPLUS               4.0 GB     disk6s2
   3:                 DOS_FAT_32 EXTRA                   10.9 GB    disk6s3
```

## Linux

You can use the command line. First, find out the device to erase:

```
lsblk
```

For this example, it will be sdc:

```
NAME         MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
(..)
sdc          179:0    0 15,8G  0 disk
└──sdc1       179:1    0 15,8G  0 part
```

Instructions:

1. Verify that the disk isn't mounted and edit the partition scheme (this step requires root privileges):

```
sudo fdisk --compatibility=dos /dev/sdc
```

```
Welcome to fdisk
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1): 1
First sector (62-31116288, default 62): 128
Last sector, +/-sectors or +/-size{K,M,G,T,P} (128-31116288, default 31116288):
2017152

Created a new partition 1 of type 'Linux'

Command (m for help): t
Selected partition 1
Hex code (type L to list all codes): 7f
Changed type of partition 'Linux' to 'unknown'.

Command (m for help): n
Partition type
   p   primary (1 primary, 0 extended, 3 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (2-4, default 2):
First sector (45-31116288, default 45): 2017280     .
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2017153-31116288, default 31116288):
7812504

Created a new partition 2 of type 'Linux'

Command (m for help): t
Partition number (1,2, default 2): 2
Hex code (type L to list all codes): 6

Changed type of partition 'Linux' to 'FAT16'.

Command (m for help): a
Partition number (1,2, default 2): 2
```

```
The bootable flag on partition 2 is enabled now.

Command (m for help): n
Partition type
   p   primary (1 primary, 0 extended, 3 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (2-4, default 3): 3
First sector (45-31116288, default 45): 9829784     .
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2017153-31116288, default 31116288):
31116288

Created a new partition 2 of type 'Linux'

Command (m for help): t
Partition number (1,2, default 2): 2
Hex code (type L to list all codes): b

Changed type of partition 'Linux' to 'W95 FAT32'.

Command (m for help): p
Disk /dev/sda
Disklabel type: dos
Disk identifier

Device     Boot   Start     End  Sectors   Size Id Type
/dev/sda1            128 2017152  2017025 984,9M 7f unknown
/dev/sda2  *    2017280 7626751  7812504   2,7G  b FAT16
/dev/sda3       9829784 7626751 21286504    21G  b W95 FAT32
```

2. Format both FAT partitions (requires root privileges)

```
sudo mkfs.fat -F 16 -n ZXUNOPLUS -s 128 /dev/sdc2
sudo mkfs.fat -F 32 -n EXTRA -s 128 /dev/sdc3
```

3. Confirm that the partition scheme has been changed:

```
lsblk
```

```
NAME      MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
(...)
sda       179:0    0 15,8G  0 disk
 ├──sda1   179:1    0    1G  0 part
 ├──sda2   179:2    0    4G  0 part
 ├──sda3   179:3    0 10,8G  0 part
```

### +3e

Once the SD card is ready to use, you can start Spectrum core with a +3e ROM and format the IDEDOS part.

The first step is determine the disk geometry. With the cart inserted into the ZXUNO+, type the command:

```
CAT TAB
```

This will give a result showing the number of cylinders, heads and sectors.

Whith this info, we estimate the size of our partition, using cylinders. For example, if the number of cylinders is 32768, and we want to use 1GB of a 16GB card, the number of cylinders needes would be 32768/16=2048. This way, the IDEDOS partition can be formatted using that number:

```
FORMAT TO 0,100,2048
```

The first value (0) is the drive to use (the first one), the second value is the maximum number of IDEDOS partitions, and the third one yis the number of cylinders to use.

Once formatted, you can create new partitions. For example, to create a 16MB partition with the name "Software", another 4GB partition named "Swap" (to use as swap) and another one name "Utils", 8MB in size:

```
NEW DATA "Software",16
NEW EXP "Swap1",4
NEW DATA "Utils",8
```

For more information about the different +3e disk commands , you can check this page at World of Spectrum.

# esxdos Commands

## Basic Guide

There are two different kind of esxdos commands, the so-called "DOT" commands, which, as the name suggests, begin with a period, and the commands that are extensions to the existing ones in BASIC.

The main "DOT" commands are the following:

- `128`: To enter 128K mode from within 48K mode
- `cd`: Change current working directory
- `chmod`: Change file attributes
- `cp`: Copy a file
- `divideo`: Play a DivIDEo (.DVO) video file
- `drives`: Show currently available drives
- `dskprobe`: Utility which shows low level content of an storage device
- `dumpmem`: Can dump RAM memory content to a file
- `file`: Tries to recognize the type of data contained in a file (like the UNIX command)
- `gramon`: Monitor to search graphics, sprites, fonts, etc. in RAM memory
- `hexdump`: Shows the contents of a file using hexadecimal notation
- `hexview`: Allow to see and navigate through the contents os a file using hexadecimal notation
- `launcher`: Creates a shortcut (launcher) to open directly a TAP or BAS file
- `ls`: Show the content of a directory
- `lstap`: Show the content of a .TAP file
- `mkdir`: Create a directory
- `mktrd`: Create a .TRD disk file
- `more`: Show the content of a text file
- `mv`: Move a file
- `partinfo`: Show partition information of an storage device
- `playpt3`: Play .PT3 music file
- `playsqt`: Play .SQT music file
- `playstc`: Play .STC music file
- `playtfm`: Play .TFC music file
- `playwav`: Play .WAV audio file
- `rm`: Remove a file or a directory
- `snapload`: Load snapshot file

- `speakcz`: Reads text aloud using czech pronunciation
- `tapein`: Mounts a .TAP file so that it can be used then from BASIC using LOAD sentence
- `tapeout`: Mount a .TAP file so that it can be used then from BASIC using SAVE sentence
- `vdisk`: Mount a .TRD disk file to use with the TR-DOS environment (once all the drives have been mounted, you can enter TR-DOS emulation by typing: `RANDOMIZE USR 15616`)

Some BASIC extended commands are:

- `GO TO` to change the current drive and/or directory (e.g.: `GO TO hd1` or `GO TO hd0"games"`)
- `CAT` to show the content of a drive
- `LOAD` to lad a file from a drive (BASIC Program, SCREEN, CODE, etc. for example `LOAD *"Screen.scr" SCREEN$`)
- `SAVE` to save data in a file (e.g: `SAVE *"Program.bas"`)
- `ERASE` to delete a file

In addition, esxdos also has an NMI manager, an application that loads when NMI is pressed, and lets you browse the SD card and load easily files (TAP, Z80, TRD, etc.). Pressing the "H" key invokes a help screen, which shows all the available keys.

The esxdos manager shows file and directory entries in the order stored in the internal FAT table, and not alphabetically. If you want to see them ordered, you have to reorder the SD card structure with a utility like Fat Sorter for Windows, FATsort for Linux and macOS, YAFS, SDSorter or other.

There are several alternative file browsers like Long Filename Browser by Bob Fossil o New NMI Handler by Dr. Slump with features that the original esxdos manager does not have

## ZXUNO+ Commands

As explained in the installation part, there are a series of commands that are exclusive to ZXUNO+:

- `back16m`: Dumps to a `FLASH.ZX1` file, in the root directory of the SD card, the contents of a 16 Meg SPI Flash memory. It must be run while using a "root" mode ROM. After finishing, it is necessary to execute the command `.ls` so that the cache is written to the card
- `corebios`: To upddate simultaneously ZX Spectrum core and BIOS
- `dmaplayw`: Plays .WAV file, which has to be 8 bits, unsigned and sampled at 15625 Hz
- `esprst`: Resets the WiFi ESP8266(ESP-12) module
- `iwconfig`: To configure the WiFi module
- `joyconf`: Configuration and tests for keyboard and DB joysticks
- `keymap`: Used to load a different keyboard map definition
- `loadpzx`: To load a .PZX tape file
- `playmid`: Plays .MID music files using the MIDI addon
- `playrmov`: Plays radastanian format video files `.RDM`). This command does not work on 48K mode.
- `romsback`: Dumps to a RomPack File named `ROMS.ZX1`, in the root directory of the SD card, all ZX Spectrum core ROMS which are stored in SPI flash memory. It must be run while using a "root" mode ROM.
- `romsupgr`: Load from a RomPack filel named `ROMS.ZX1`, in the root directory of the SD card, all ZX Spectrum core ROMS into SPI flash memory. It must be run while using a "root" mode ROM
- `upgr16m`: Load the content of a `FLASH.ZX1` file, in the root directory of the SD card, to a 16 Meg SPI Flash memory. It must be run while using a "root" mode ROM
- `zxuc`: Utility to configure al options of BIOS, which also can be stored in the SD in configuration files that can be loaded later
- `zxunocfg`: Configuration utillity for certain features of ZX-Uno such as timings, contention, keyboard type, CPU speed, video type or vertical frequency

# Wi-Fi

The +UNO has an ESP module with a Wi-Fi chip ESP8266. It can easily be used with a ZX Spectrum core (e.g., EXP27 160820 core) which has synthesized an UART device, that allows communication with the module.

There are two "DOT" commands for configuring software access to the module. They can be downloaded from GitHub official repository:

- `esprst`, which restarts the module
- `iwconfig`, to register the Wi-Fi network name (SSID) and password, keeping them in the file `/sys/config/iw.cfg`.

For example:

```
.iwconfig mywifi mypassword
```

Since the selected VGA frequency affects the master clock frequency, for the communication between the core and the Wi-Fi module to work correctly, it has to be set at 50 (see the BIOS settings chapter).

## Network tools for ZX-Uno pack

These are programs, developed by Nihirash and that are available to download from his web.

- `netman`: Utility to configure the ESP Wi-Fi chip for other programs from Nihirash. Does not work in 48K mode
- `uGophy`: Gopher client. Does not work in 48K mode
- `irc`: Internet Relay Chat client. Works better at 14 Mhz
- `wget`: Utility to download files with HTTP (does not work with HTTPS)
- `platoUNO`: PLATO client. Also works better at 14 Mhz. For more information about PLATO, check IRATA.ONLINE web

## FTP-Uno

FTP cliente developed by Yombo, available at GitHub.

Configuration steps:

1. Edit `FTP.CFG` file with all the required information (SSID and password, FTP server, etc.)
2. Copy `FTP.CFG` inside `/SYS/CONFIG/` in microSD card
3. Also copy `ftpUno.tap` to any place in the card
4. Start up the ZX-Uno and load the tape file `ftpUno.tap`

## UART Terminal

Program example included with ZXYLib C library, developed by yombo, that let's you send directly typed characters using the UART, and also see the result. Available to download at this link.

Once the file `UARTTERM.tap` is in the card and loaded, you can type several specific commands for ESP8266 chip. For example:

- `AT`. To check if ther is communication. `OK` would be the result if everything is fine
- `AT+RST`. To restart the chip. Exactly what `esprst` command does
- `AT+GMR`. To see some information, like firmware version, etc.
- `AT+CWMODE_CUR=1`. Put temporarily the chip into Wi-Fi client mode, until next restart
- `AT+CWMODE_DEF=1`. Put temporarily the chip into Wi-Fi client mode, and save it as default
- `AT+CWJAP_CUR="<WiFiNetwork>","<WiFiPassword>"`, where `<WiFiNetwork>` Wi-Fi ID of the network to connect to, and `<WiFiPassword>` the access password, connects temporarily to that network
- `AT+CWJAP_DEF="<WiFiNetwork>","<WiFiPassword>"`, connects to the network, and saves the settings as default in the chip flash memory
- `AT+CWAUTOCONN=1` sets the chip to connect automatically on boot to the default network (`AT+CWAUTOCONN=0` disables it)

You can see all the available commands reading the official documentation.

# Making RDM (RaDastan Movie) files

The PLAYRMOV "DOT" command plays radastanian format video files. To convert your own videos, you need makevideoradas, a utility that is available at SVN repository.

If using Windows, there is already an executable file (makevideoras.exe). For Linux or macOS, you must have installed command line developer utilities in order to compile an executable

```
gcc makevideoradas.c -o makevideoradas
```

Apart from makevdideoradas, you need another two tools: ffmpeg and imagemagick. These can be installed with a package manager (apt, yum, pacmam, brew, etc.) or downloading the source code and compiling.

Now, the first step to convert our video (for example myvideo.mp4), is exporting the frames as 128x96 pixel BMP image files. We create a temporary file (img for this example), to store them.

```
mkdir img
(...)/ffmpeg -i myvideo.mp4 -vf "scale=128:96,fps=25" -start_number 0
img/output%05d.bmp
```

Now we transform the BMP files to 16 colours (v3) BMP files.

```
(...)/magick mogrify -colors 16 -format bmp -define bmp:format=bmp3 img/*.bmp
```

Finally, we assemble the .RDM file (in this example myvideo.rdm) and cleanup the temporary files and directory.

```
(...)/makevideoradas img/output
mv img/output.rdm ../myvideo.rdm
rm -rf img
```

There is more information about all this process at this thread in Zona de Pruebas forums.

# CP/M

There's a Multicomp core for ZX-Uno, which is CP/M 2.2, or you can run CP/M 3.0 for Spectrum with a +3e ROM and a specially prepared SD card. However, only 51 columns are available, and you will have to be jumping between the left and the right side in order to see all the 80 columns, which is quite cumbersome.

Multicomp

multicomp_z80_CPM.ZX1 https://obsolescence.wixsite.com/obsolescence/multicomp-fpga-cpm-demo-disk http://searle.x10host.com/Multicomp/index.html http://searle.x10host.com/Multicomp/cpm/fpgaCPM.html#UsingTheMachine

+3

https://www.zxuno.com/forum/viewtopic.php?f=12&t=1427&hilit=core#p16169
https://www.zxuno.com/forum/viewtopic.php?f=39&t=4099

# FUZIX

FUZIX is a fusion of various elements from UZI (an implementation of the Unix kernel written for a Z80 based computer), extended from the 7th Edition Unix kernel to somewhere in the SYS3 to SYS5.x world, with bits of POSIX.

It is not yet useful although you can build and boot it and run test application code. A lot of work is still needed on the utilities and libraries.

At the moment of writing these lines, the officially built images do not work with all the ZXUNO+ ZX Spectrum cores. However, building from the source code, does work. The following instructions have been tested with the latest code on June 2021.

## How to Build

The following instructions have been made using a clean installation of Fedora Workstation Linux (Fedora 34). Apart from the package installation commands, all the other steps should work with many other Linux distributions.

Install the needed packages:

```
sudo dnf groupinstall -y 'Development Tools'
sudo dnf install -y gcc-c++ automake boost-devel gputils flex texinfo bison byacc
```

Get the special version of SDCC compiler for Fuzix, and build it:

```
git clone https://github.com/EtchedPixels/sdcc280.git

cd sdcc280
cd sdcc
./configure
make
sudo make install
cd ../..
```

Get Fuzix source code:

```
git clone https://github.com/EtchedPixels/FUZIX.git
cd FUZIX
```

Edit `Makefile` and change the line with `TARGET=` to `TARGET=zxdiv`. Build:

```
sudo make
```

Get the esxdos binary and kernel image from this paths:

```
./Kernel/platform-zxdiv/FUZIX
./Kernel/platform-zxdiv/FUZIX.BIN
```

Build the root filesystem:

```
cd ./Standalone/filesystem-src
./build-filesystem rootfs 256 65535
cd ../..
```

Get the root filesystem image file from this location:

```
./Standalone/filesystem-src/rootfs
```

## How to use

You need a MBR partition table on the SD card. You can set up one or two primary partitions as usual (the first one with a functional esxdos installation), leaving enough space at the end to add one 32MB (Type `7E`) primary partition for the root file system and one 4MB (Type `7F`) primary partition for swap.

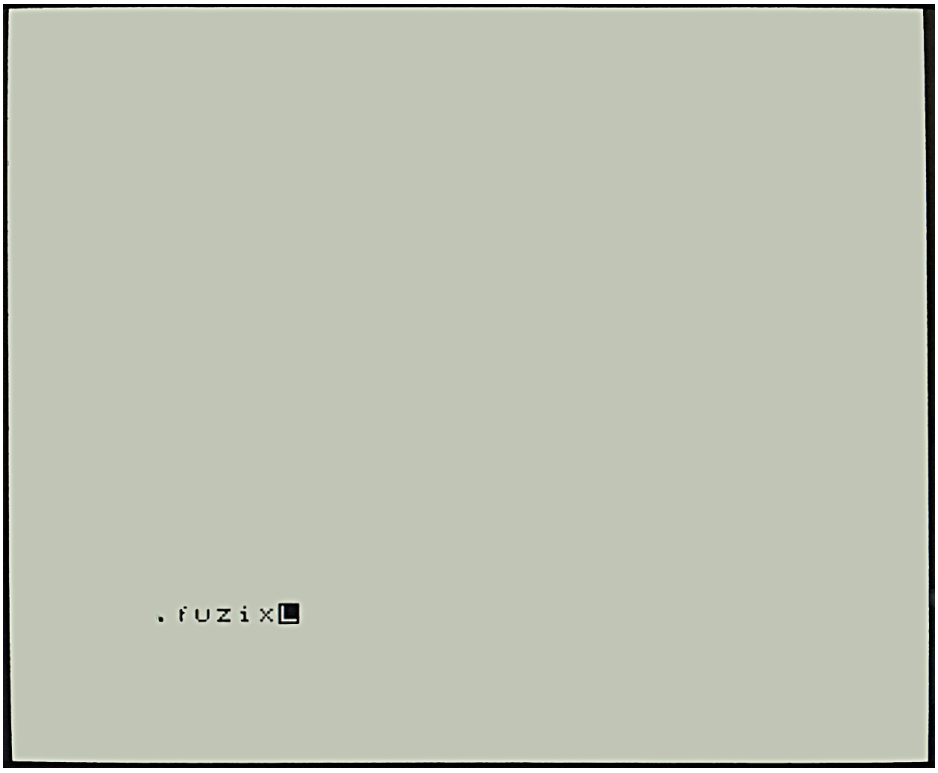
Copy the rootfs filesystem to the type `7E` partition. You can use the `dd` utility, included with Linux, macOS, etc. (and also ported to Windows).

After you find the device name for the `7E` partition, use that as destination for the `rootfs` file. For example, for `/dev/rdisks3`:

```
sudo dd if=rootfs of=/dev/rdisks3
```

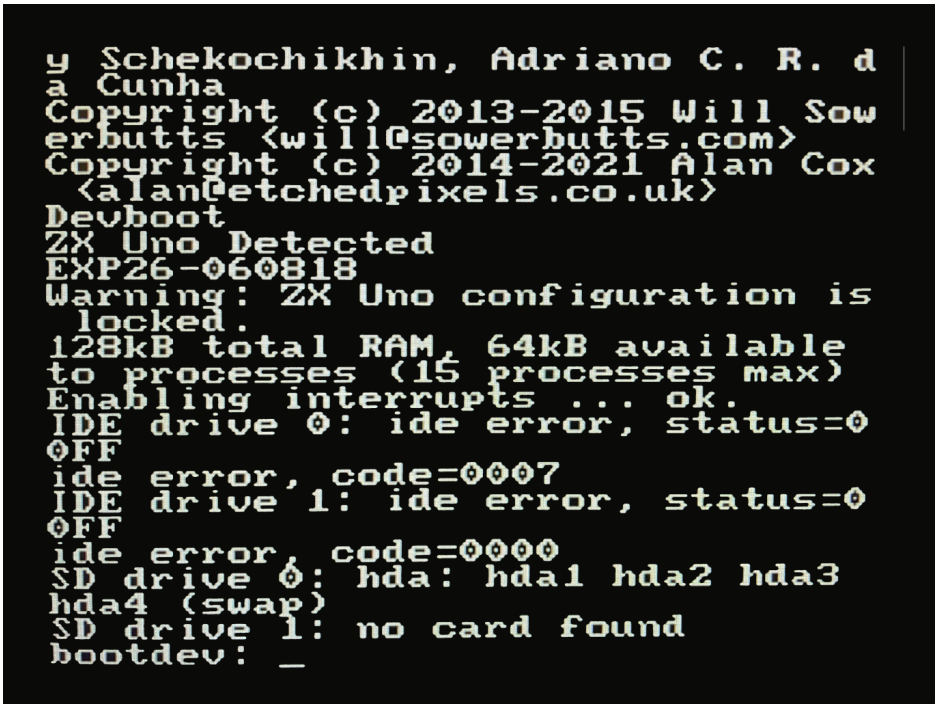
Copy the `FUZIX` command into the `BIN` directory and copy `FUZIX.BIN` to the top level directory of the esxdos partition.

Boot into a Spectrum core with a 128K ROM and with esxdos, then type '.fuzix', and press Enter.

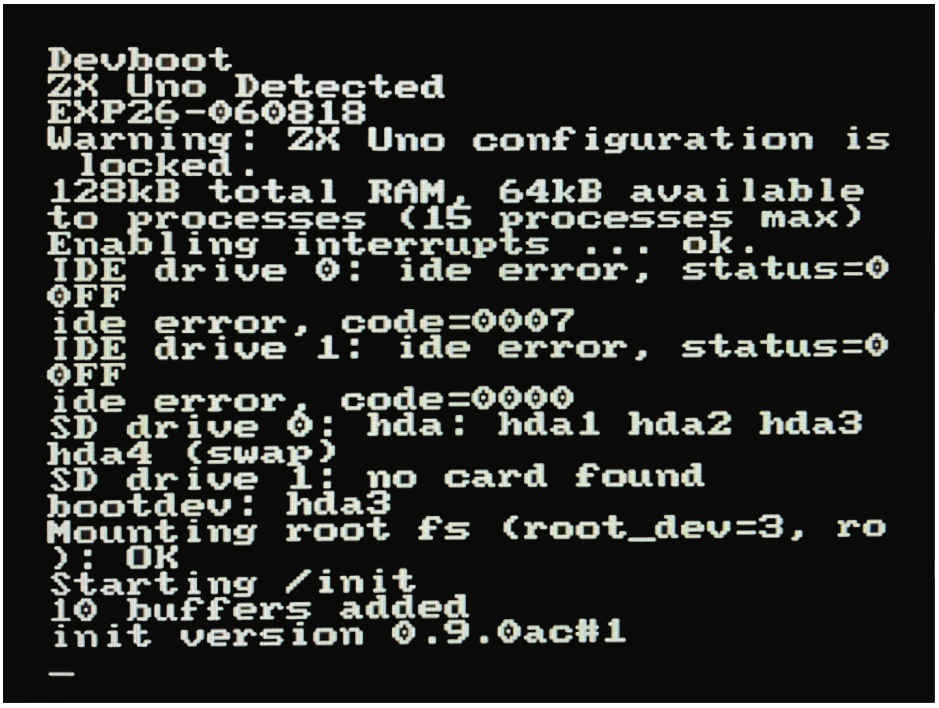



Your keyboard configuration on BIOS should be using an english layout, or you won't be able to type some characters like |.

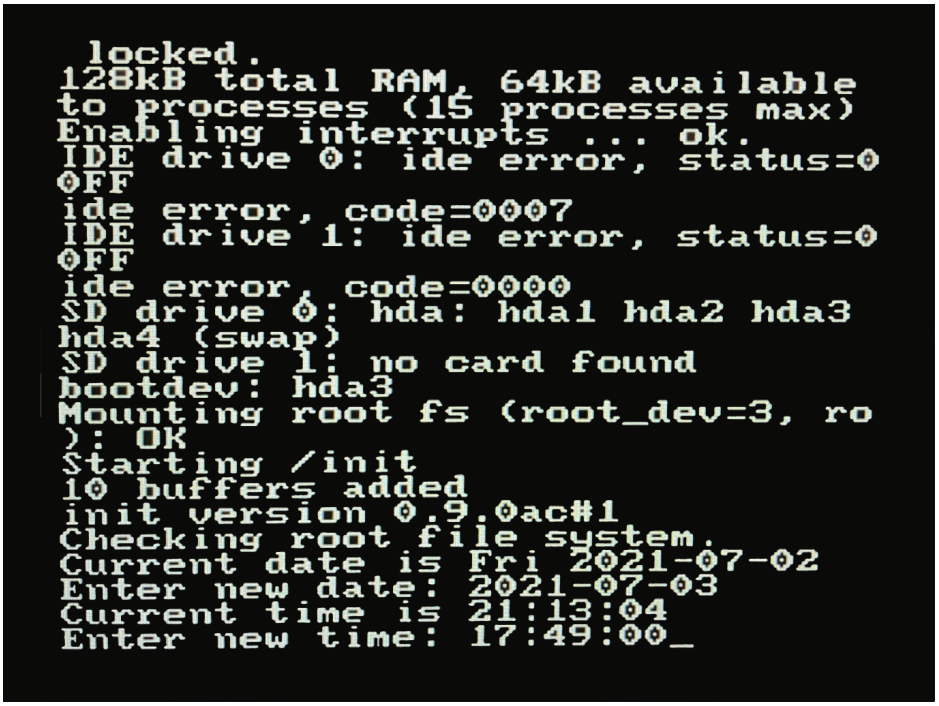
After a few seconds, the system should detect the SD card and find the partitions. In this example, the root is the third partition of SD 0.



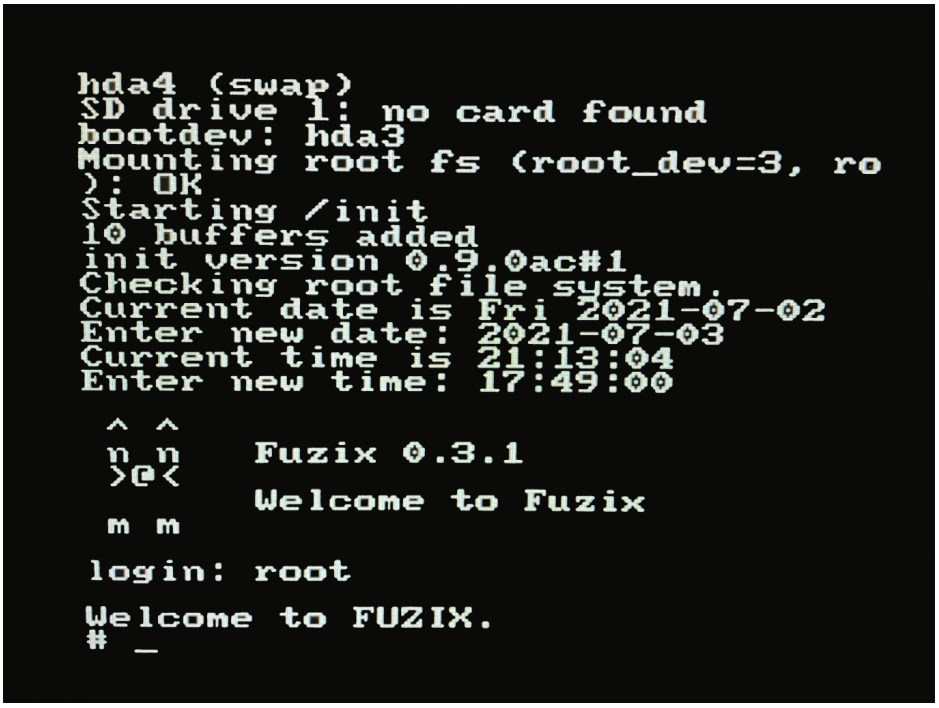
Type the rootfs partition (e.g. hd3) and press Enter.




Set up date (press Enter) and time (press Enter).



Login with `root` user and no password.




Now you have a Fuzix shell.

When finished, remember to stop the system using the `shutdown` command or the root filesystem will be marked as not clean, and a filesystem check will be forced on the next Fuzix boot.

# Upgrade

## BIOS

To update the BIOS, a file named `FIRMWARE.ZX1` (for a ZXUNO+ with an FPGA LX16 board) or `FIRMWARE.ZX1` must be obtained. The latest version of the firmware files can be downloaded from the official repository

Updating the firmware (BIOS) is delicate. It should not be done if it is not necessary. If doing so, ensure that the ZXUNO+ has uninterrupted power (such as a UPS or a laptop USB with battery).

Copy the file to the root of the SD card, turn on and press `F2` to enter BIOS, select `Upgrade`, choose *"Upgrade BIOS for ZX"*, and then *"SDfile"*. The system will read the file `FIRMWARE···` and notify when finished.

## ROMs

The flash memory of a ZXUNO+ has reserved 64 slots, 16K each, to store ZX Spectrum ROM images. Thus, an original ZX Spectrum ROM (16K) will take one slot, a ZX Spectrum 128K ROM (32K) will be two slots, and a ZX Spectrum +2A ROM (64K) will need 4 slots.

You can add a new ROM pressing the key `N` at the BIOS ROMs screen, connecting an audio cable to the board, and playing a ROM audio tape. ROM audio tapes can be made from a `.tap` file built with the `GenRom` utility, available at ZX-Uno Code Repository.

To update at once all the ROMs installed for ZX Spectrum, a RomPack file named `ROMS.ZX1` must be obtained, which must be copied to the SD card. Boot the ZXUNO+ using a "rooted" ROM, and then just enter the command `.romsupgr`. This will burn all the ROMs, which will be available for use.

Remember that if the ZXUNO+ is started while pressing the / key, then the default ROM of the ZX Spectrum core will be loaded in" root "mode.

To do the opposite process (save the ROMs in a RomPack file named `ROMS.ZX1`), you can use the` .romsback` command.

RomPack files can be easily edited with the ZX1RomPack utility. Although it is a Windows program, it works perfectly, for example using Wine or similar programs, either on macOS or Linux.

# Cores

There are a number of available spaces where you can store cores (the number depends on the size of the SPI Flash of the ZXUNO+ model), the first space being reserved for the main ZX Spectrum (this does not prevent having more ZX Spectrum cores in other space as well of the first).

Official cores are available to download from GitHub repository.

To update or install a new core there are several possibilities.

The easiest way is to obtain the latest version of the file that defines the core, which will be a file that must be named `COREnn.ZX1`, where `nn` is the slot number where to install (for example `CORE.ZX1` or `CORE2.ZX1` for slot 2).

Starting with BIOS version 0.80, files are named using the `COREXXy.ZXn` convention where XX *always* is a two-digit number. Thus, an old `CORE4.ZX1` file has to be renamed as `CORE04.ZX1`. The `y` part of the name is ignored, so longer and more descriptive names can be used (such as `CORE04_example.ZX1`).

Copy the file to the root of the SD card, turn on and press `F2` to enter BIOS. Choose `Upgrade`, select the row corresponding to the chosen core number (for example, 2 - just after Spectrum), press enter and then *" SD file "*. The system will read the file `COREnn ..` and warn when it is updated, although first it will ask for the name (to be shown in the list to choose from at startup and in the BIOS list).

The ZX Spectrum core update is exactly the same as other cores, but instead of the name `CORE1.ZX1`, it has to be a file named `SPECTRUM.ZX1`.

# esxdos

To update esxdos to a new version, the distribution must be obtained from the official website.

Once downloaded and extracted, the contents of `BIN` and `SYS` directories have to be copied to the root of the card, merging the existing ones (to preserve the exclusive ZXUNO+ commands).

Copy `ESXMMC.BIN` (or `ESXMMC.ROM`, depending on version) to the root of the SD card.

Start ZXUNO+ with the card inserted and press `F2` to access BIOS setup. Select the `Upgrade` menu and choose *"Upgrade esxdos for ZX"*. In the dialog that appears choose *"SD file"* and, when it asks *"Load from SD"* answer *"Yes"* to the question *"Are you sure?"*. The content of the file `ESXMMC⋯` will be read, written to the flash storage and you will be notified when it is updated.

Do a Hard-reset, or turn it off and on.

If everything has been done correctly, when you turn on the ZXUNO+ you will see how esxdos detects the card and loads the necessary components to work, showing the new version at the top.

# Flash Memory

You also can update all the FPGA flash memory. At this moment, from the BIOS you can only use 16MiB image files.

Copy the image file (16MiB) `FLASH.ZX1` to the root of the SD card.

Turn on the ZXUNO+ and press the `F2` key during boot to access the BIOS setup. Select the menu `Upgrade` and then choos the option *"Upgrade flash from SD"*. Press Enter, choose `Yes`, and press Enter again to start the Flash writing process.

Do a Hard-Reset or turn of and on again.

This process can't be undone, and it will replace all the previously installed cores, the BIOS, the ZX Spectrum ROMs and their configuration with the data in the image file.

# Other cores

## ZX Spectrum EXP27

Based on the source of EXP27-160820 core by mcleod_ideafix, with some modifications (by Spark2k06, Neuro, Yombo and azesmbog):

- Different colour modes including monochrome
- Option to enable or disable the Wi-Fi addon
- Modified activation of Turbo Boost Mode
- Gest back the option to use custom marks in PZX, just like the first T24 unofficial core did (without SAA1099)
- SAA1099 audio chip (without PZX)

There are two differente versions of the core, one with PZX support and no SAA1099 audio chip, and another one with SAA1099 chip but without PZX support.

### SD card format

You may use a SD card with the first partition formatted as FAT16 or FAT32, and inside, an esxdos distribution matching the version installed in BIOS (see the esxdos corresponding section for more info).

See the corresponding section for instructions of how to install this alternative core in ZXUNO+.

## Keyboard

### Special keys and buttons

While the core is running:

- `End`: Switches between colour, green monochrome, amber monochrome and black and white
- `F6` (`Caps Shift+Symbol Shift+6` on +UNO): Sets up a custom mark at the current PZX position (only in the version with PZX support)
- `F7: PXC Play/Pause (only in the version with PZX support)
- `F8: Moves the PZX position to a the custom mark (only in the version with PZX support)
- `CTRL+ F8: Rewinds the PZX position to a "tag BRWS" mark. These marks are used, for example, to label where a multiload starts (only in the version with PZX support)
- `F9: Stops playing the PZX and set the position back to the beginning (only in the version with PZX support)
- `F11` (`Caps Shift+Symbol Shift+Q` on +UNO): Enabe or disable the Wi-Fi module
- `F12` (`Caps Shift+Symbol Shift+W` on +UNO): Enable or disable 28 MHz CPU speed.

The PZX player will stop when it gets to the end of the file, or if it finds a STOP mark, or if it finds a STOP IF IN 48K mark and the core is in 48K mode (no 128K pagination).

You can add BRWS and STOP marks to a PZX file using the program ZX-BlockEditor, which is part of de ZX-Modules.

# ZX Spectrum 48K (Kyp)

Alternative core, whose objective is to be the most accurate implementation in timings, memory contention, etc.

Main features:

- Specdrum
- Turbosound (two AY chips) with mix selection ACB/ABC
- DivMMC with esxdos 0.8.8
- Composite video/RGB and VGA video output

## SD card format

You need a SD card with the first partition formatted as FAT16 or FAT32, and inside, the standard esxDOS 0.8.8 (see the esxdos corresponding section for more info).

See the corresponding section for instructions of how to install this alternative core in ZXUNO+.

## Keyboard

### Special keys and buttons

While the core is running:

- `Esc`: BREAK
- `F5` (`Caps Shift+Symbol Shift+5` on +UNO): NMI
- `F8` (`Caps Shift+Symbol Shift+8` on +UNO): Change Turbosound mixer configuration between ACB and ABC.
- `Scroll Lock`: Switches between composite and VGA video modes.
- `Ctrl+Alt+Backspace` or `F11`: Hard reset. Backspace is the delete key, located in the top-right portion of the keyboard, above `Enter`.
- `Ctrl+Alt+Supr` or `F12`: Soft reset.

# ZX Spectrum 128K (Kyp)

Alternative core, whose objective is to be the most accurate implementation in timings, memory contention, etc.

Main features:

- Specdrum
- Turbosound (two AY chips) with mix selection ACB/ABC
- DivMMC with esxdos 0.8.8

## SD card format

You need a SD card with the first partition formatted as FAT16 or FAT32, and inside, the standard esxDOS 0.8.8 (see esxdos corresponding section for more info).

See the corresponding section for instructions of how to install this alternative core in ZXUNO+.

## Keyboard

### Special keys and buttons

While the core is running:

- `Esc`: BREAK
- `F5` (`Caps Shift+Symbol Shift+5` on +UNO): NMI
- `F8` (`Caps Shift+Symbol Shift+8` on +UNO): Change Turbosound mixer configuration between ACB and ABC.
- `Ctrl+Alt+Backspace` or `F11`: Hard reset. Backspace is the delete key, located in the top-right portion of the keyboard, above `Enter`.
- `Ctrl+Alt+Supr` or `F12`: Soft reset.

# ZX Spectrum (Kyp/azesmbog)

A mixed version, synthesized by azesmbog, of the two cores <<,#_zx_spectrum_48k_kyp,48K>> y 128K by Kyp, adding also a Pentagon mode.

Its main features are:

- Specdrum
- Turbosound (two AY chips) with mix selection ACB/ABC
- SAA1099 chip
- DivMMC with esxdos 0.8.9
- Composite video/RGB and VGA video output at 50Hz

## SD card format

You need a SD card with the first partition formatted as FAT16 or FAT32, and inside, the standard esxDOS 0.8.8 (see esxdos corresponding section for more info).

See the corresponding section for instructions of how to install this alternative core in ZXUNO+.

## Keyboard

### Special keys and buttons

While the core is running:

- `F5` (`Caps Shift+Symbol Shift+5` on +UNO): NMI
- `F8` (`Caps Shift+Symbol Shift+8` on +UNO): Change Turbosound betwween ACB and ABC.
- `F10` (`Caps Shift+Symbol Shift+0` on +UNO): Pentagon Mode
- `F11` (`Caps Shift+Symbol Shift+Q` on +UNO): 48K Mode
- `F12` (`Caps Shift+Symbol Shift+W` on +UNO): 128K Mode
- `Ctrl+Alt+Backspace` or `F11`: Hard reset. Backspace is the delete key, located in the top-right portion of the keyboard, above `Enter`
- `Ctrl+Alt+Supr` or `F12`: Soft reset.

The default mode on startup is 128K.

To see the standard menu on 128K mode, you have to use the esxdos dot command: `.128`.

# ZX Spectrum TBBlue

The TBBlue firmware, was created by Victor Trucco and Fabio Belavenuto on en 2016. Originally thought to use a Z80 processor, it was changed afterwards to usa Xilinx Spartan-6 FPGA, to add some more advanced features. This the base for ZX Spectrum Next.

The lates ZX-Uno TBBlue core is a version synthesized by azesmbog from the latest versions of the repository and has, among other, these features:

- Composite video/RGB and VGA video output
- SD support, with DivMMC
- Turbo Mode
- Timex Mode
- ULAPlus
- Multiface support
- Joystick support
- Turbosound
- Spectrum ROM selection on boot
- Tree AY audio chips (as on ZX Spectrum Next), added by azesmbog
- SID sound chip, added by azesmbog

## SD card format

An SD card with the first partition in FAT16 or FAT32 format is needed with, optionally, the esxdos distribution that matches the current BIOS configuration (see the esxdos corresponding section for more info).

Obtain the TBBlue distribution that matches the core version, for example, looking in ZX-Uno forums. Usuaally, the structure is something like this:

```
/
+-TBBLUE/
|  +-128.rom
|  (...)
|  +-config.ini
|  (...)
|  +-esxmmc.rom
|  (...)
|
+-TBBLUE.FW
```

If you are using a SD card with esxdos, the `esxmmc.rom` file version must match the one of the SD card.

See the corresponding section for instructions of how to install the TBBlue core in ZXUNO+.

## Keyboard

### Special keys and buttons

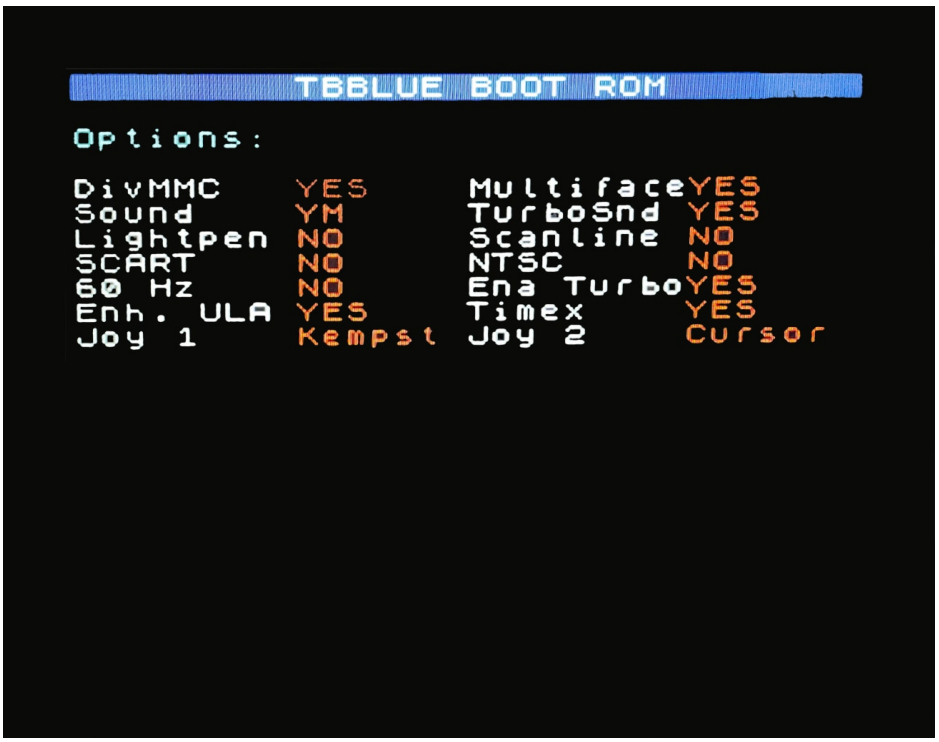
While the core is running:

F1: Hard Reset Space+F1: Boot menu F2: Enabe or disable Scandoubler F3: Switches between 50Hz and 60Hz F4: Soft reset F5: CPU Pause F9: Multiface F10: NMI

## Basic Guide

Pressing Space+F1 a core reset occurs and then the boot menu appears, where the existing boot options in the config.ini file are shown.

Then, pressing E, you can access to the other core configuration options. You can change them with the cursor keys and the Space key. Press Enter to finish.



# Acorn Atom

Acorn Atom was a home computer made by Acorn Computers Ltd. The ZXUNO+ core is an adaptation of the AtomFPGA project. You can get more information at ZX-Uno Forums.

## SD card format

You have to use a SD card with the first partition in FAT16 format.

Download the latest version of Atom Software Archive from GitHub.

You can set up the files in the SD in two different ways:

1. Extract all the contents of the archive to the root of the SD card. `SYS` directory contents are compatible with esxdos `SYS` directory, so you can merge both into one.
2. Have less files an directories in the root directory. Create a directory named `ATOM` in the SD root, and copy inside all the uncompressed archive content, except for the directory `MANPAGES` which must also be in root. Then, extract and the files from `trick_ATOM_folder` archive (available at ZX-Uno Forum), replacing any file with the same name. You will get a file and directory structure like this:

```
/
+-ATOM/
|  +-AA/
|  (...)
|  +-AGD/
|  | +-SHOW2
|  | +-SHOW3
|  (...)
|  +-MENU
|  (...)
|  +-TUBE/
|  | +-BOOT6502
|  (..)
|
+-MANPAGES/
|  +-CPM.MAN
|  +-FLEX.MAN
|  (...)
|
+-MENU
```

# Keyboard

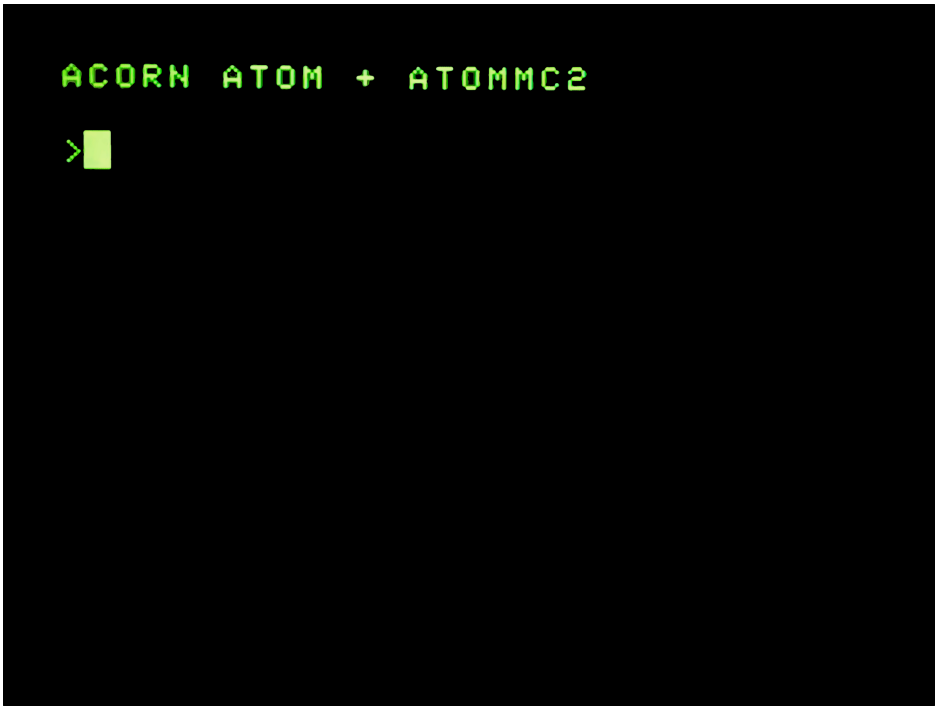
### Special keys and buttons

While the core is running:

- `Shift+F10`: Shows Atom Software Archive Menu
- `F10` (`Caps Shift+Symbol Shift+0` on +UNO): Soft Reset
- `F1` (`Caps Shift+Symbol Shift+1` on +UNO): Turbo mode 1Mhz
- `F2` (`Caps Shift+Symbol Shift+2` on +UNO): Turbo mode 2Mhz
- `F3` (`Caps Shift+Symbol Shift+3` on +UNO): Turbo mode 4Mhz
- `F4` (`Caps Shift+Symbol Shift+4` on +UNO): Turbo mode 8Mhz

The keyboard uses the following mapping:

## Basic Guide




Once it's running, press `Shift+F10`, or type `*MENU` and `Enter`, to show a menu where you can choose and load Atom Software Archive programs from the card.

# Acorn Electron

The Acorn Electron was a budget version of the BBC Micro educational/home computer. The core is based on David Banks (hoglet) original works.

Main features:

- Composite video/RGB and VGA 50Hz video output
- SD support with ".MMB" files
- Software loading via ZXUNO+ audio in port
- PS/2 keyboard

## SD card format

An SD card with the first partition in FAT16 or FAT32 format is needed to load software from it. A special ROM inside the core (Smart SPI) reads a special file with disk images inside.

The file must have the name `BEEB.MMB` and it has to be in the root directory. You can make one with `MMBImager` for Windows, available at ZX-Uno SVN Repository (Usuario `guest`, contraseña `zxuno`) or with MMB/SSD Utils in perl, available at GitHub.

The file has to be whole across the SD (not fragmented). You can use a program that can defrag files or FAT filesystems (like Defraggler for Windows) or use the following method:

1. Format the first SD partition using FAT16 or FAT32, but **NOT with quick format** (when using Windos, uncheck that option).
2. Copy `BEEM.MB` making sure it is the **FIRST file** being copied.
3. If you want you can add any other file to the SD (e.g. for using with other cores), but keep **ALWAYS** `BEEB.MMB` as the first file copied to the card.

Since the `MMB` file has the same name as the one for BBC Micro core, you can use another core with full access to the SD card (like a a ZX Spectrum core with esxdos), with both `MMB` files with a different name, and then rename the one for Acorn Electron as `BEEB.MMB` before running this core

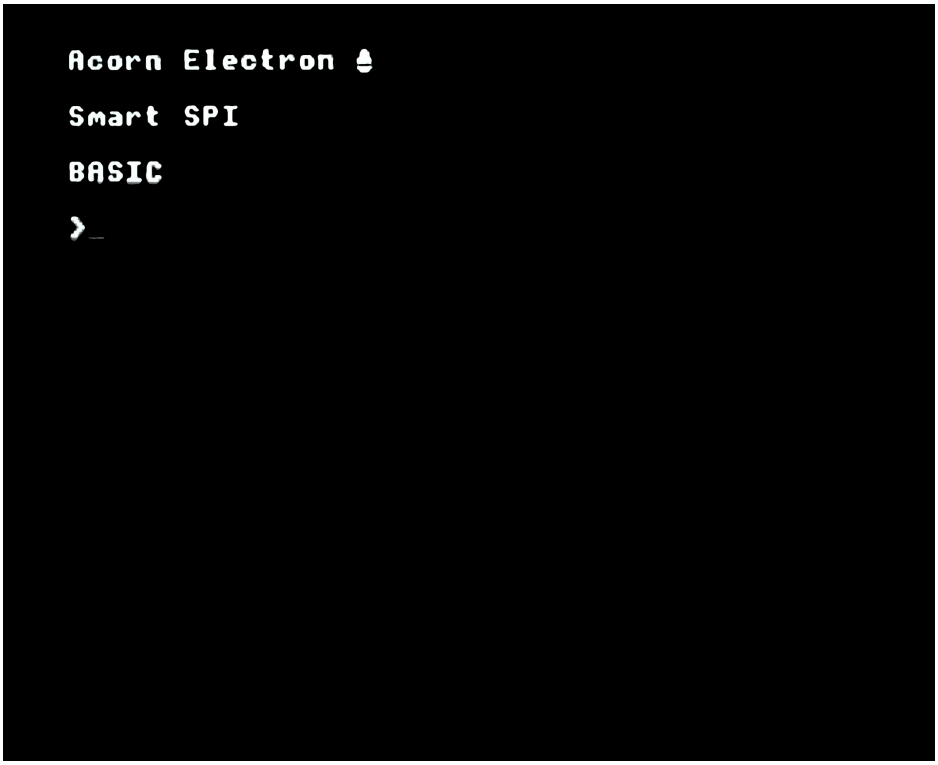
## Keyboard

### Special keys and buttons

While the core is running:

- `Scroll Lock`: Switches between composite and VGA video.
- `Ctrl+Shift+1` to `Ctrl+Shift+4`: Use other graphic modes (60Hz, etc)
- `F10` and `Ctrl+F10`: Soft Reset
- `Ctrl+Alt+Backspace` (`Caps Shift+Symbol Shift+B` on +UNO): Hard reset. Backspace is the delete key, located in the top-right portion of the keyboard, above `Enter`

## Basic Guide




Once the SD card is inserted and the core running, if `BEEB.MMB` file was created correctly, on start, you should see:

```
Acorn Electron
Smart SPI
BASIC
>
```

Disc 0 from the image file is mounted automatically, and you can see it's contents using the command:

```
*CAT
```

To load, for example, the menu available with some image files available on internet, use the command:

```
CHAIN"MENU"
```

To load using the audio input:

```
*TAPE
CHAIN""
```

And then start playing the external audio device.

To show the list of available discs inside `BEEB.MMB` file:

```
*DCAT
```

To put a particular virtual disc in a virtual drive:

```
*DIN discnum drivenum
```

# Amstrad CPC 464

The Amstrad CPC 464 was the first of a series of 8-bit home computers produced by Amstrad.

ZXUNO+ version has been made by McLeod.

Core features:

- Full Amstrad CPC 464: 64KB RAM, 32KB ROM, tape interface, keyboard and joystick
- RGB/composite video and VGA (50Hz) support
- VGA Scanlines
- 1 player joystick support

There is another version (Amstrad CPC6128) of this core, synthesized by Jepalza, which adds 128K memory pagination, It would be like a CPC6128 with a broken floppy drive

## SD card format

This core does not use the SD card.

## Keyboard

### Special keys and buttons

During core execution:

- `Del`: `CLR`.
- `Print Scr` or `Left Windows`: `COPY`
- `F10` and `Ctrl+F10`: Soft Reset.
- `Ctrl+Alt+F5`: NMI.
- `Ctrl+Alt+Del` (`Caps Shift+Symbol Shift+N` on +UNO): Reset.
- `Ctrl+Alt+Backspace` (`Caps Shift+Symbol Shift+B` on +UNO): Hard reset. Backspace is the delete key, located in the top-right portion of the keyboard, above `Enter`.
- `End`: Switches beteween color and green screen modes

## Basic Guide


When using BASIC, you can load a external tape (or other external audio device) with the command `RUN"`. Unlike the original machine, you can hear the audio while playing the tape.

# Amstrad CPC 6128

The Amstrad CPC 6128 was the successor to the Amstrad CPC 664 (only produced for approximately six months), and this one, was the sucessor to the Amstrad CPC 464.

ZXUNO+ Amstrad CPC 6128 core is based on the FPGAmstrad project by Renaud Hélias.

Some of its features are:

- VGA: 640x480 VGA centered at 60Hz
- Disk selection: The first disk image detected is inserted on startup, and pressing a key makes a reset and loads the next one

## SD card format

You have to use a SD card with the first partition in FAT32 format (`0B` Win95 FAT-32 Partition Type), with a maximum of 4GB in size, and 4096 bytes per cluster.

You also need the following ROM files (they are available at the original project Wiki) or from the GitHub repository:

- `OS6128.ROM`
- `BASIC1-1.ROM`
- `AMSDOS.ROM`
- `MAXAM.ROM`

It is also recommended to copy one or more disk image files (`DSK`) with the software that you want to run.

Copy all `ROM` and `DSK` files to the root directory of the FAT32 partition.

## Keyboard

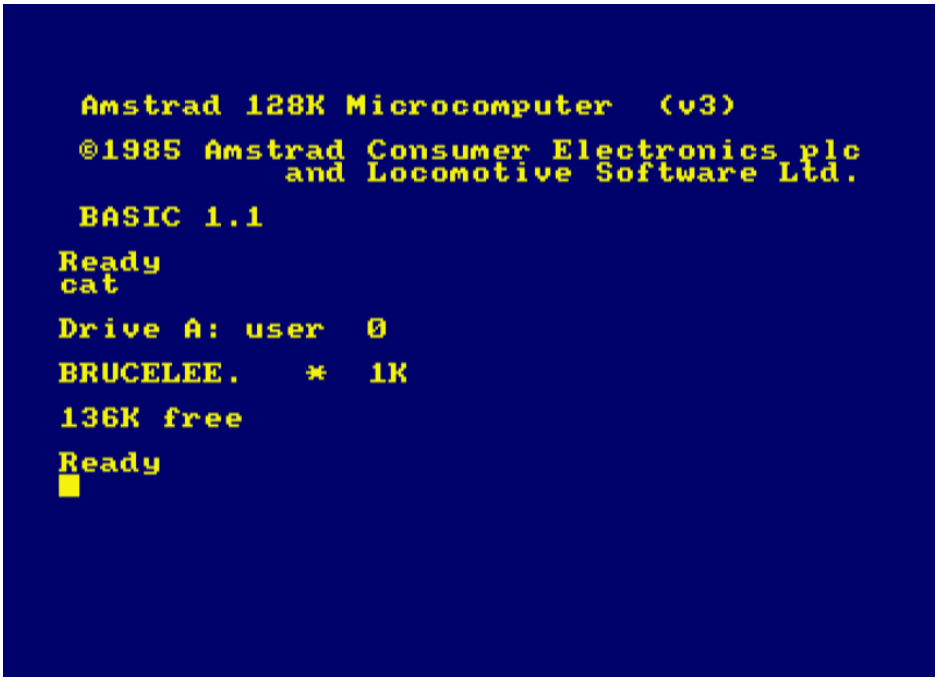
### Special keys and buttons

During core execution:

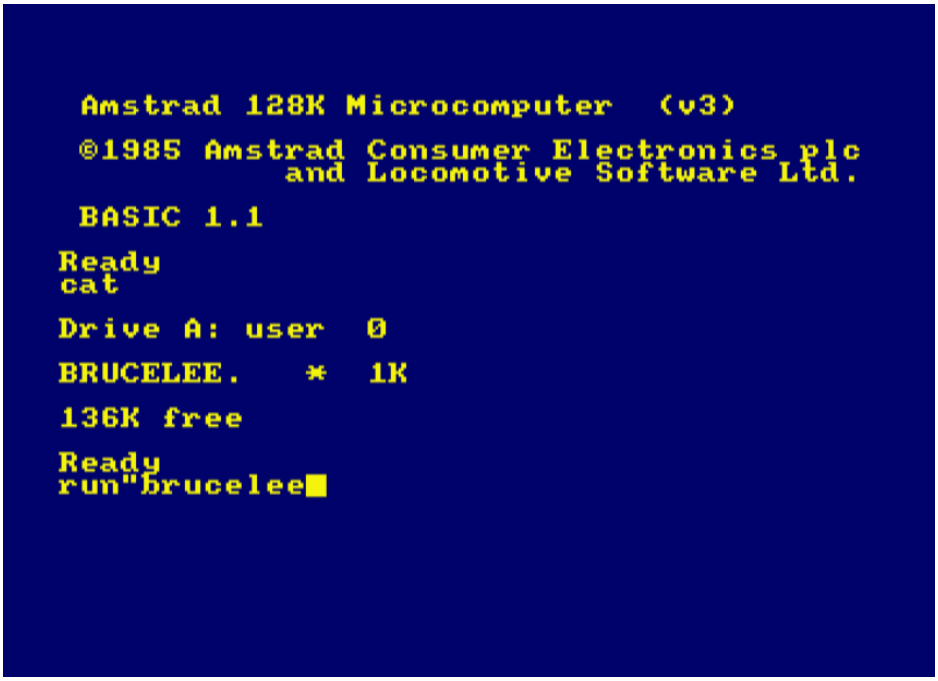
- `Page Up`: Reset the Amstrad computer and load the next `DSK` file alphabetically
- On a PS/2 keyboard, only the left shift key works properly

## Basic Guide

Use the CAT command to see the contents of the currently loaded DSK file.




Type the command RUN"<name> to load a program from disk




Press Page Up key to reset and load the next DSK file.

# Apple I

The Apple I, or Apple-1, was one of the first desktop computers, and te first to combine a microprocessor with a keyboard connection and a monitor.

The ZX-Uno core has been made by Subcritical, using the Apple-One project as a basis.

Som of its features are:

- PS/2 keyboard (english layout)
- Only VGA output
- Integer Basic included

## SD card format

This core does not use the SD card.

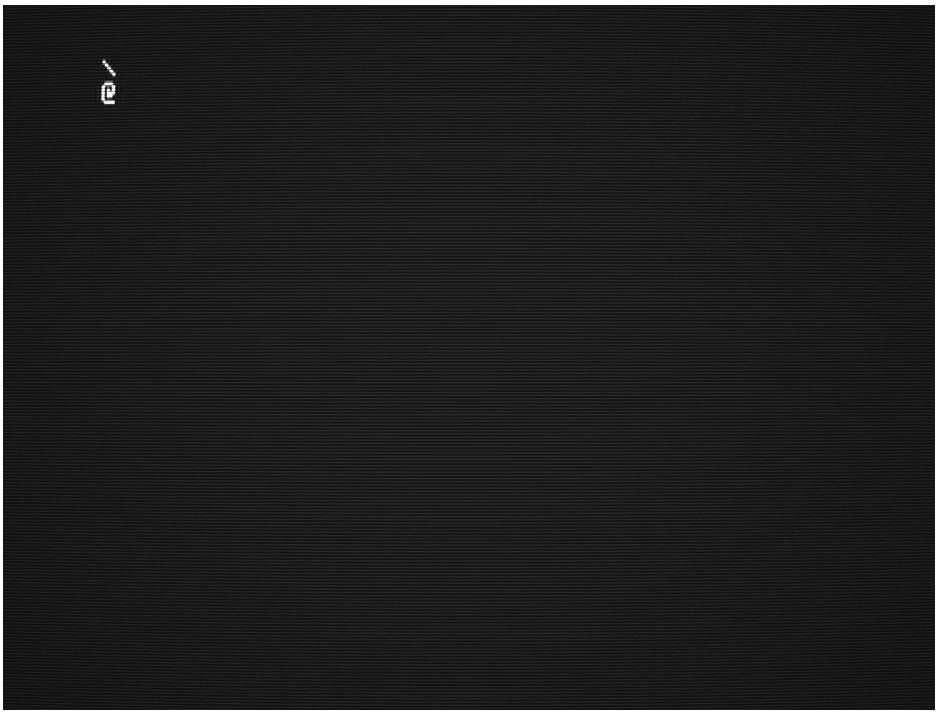
## Keyboard

The keyboard uses an english layout.

Take note that, on an Apple I, there is no delete key. Due to the way that the video signal is made, it is not possible to make the cursor come back to delete a character, so instead, a special one is used: `_`. This means that the text has gone back to delete from the data input (for example `PRN_INT` is interpreted as `PRINT`).

## Basic Guide

Upon start, the Apple I boots with Woz Monitor (ROM loaded at `0xFF00`)

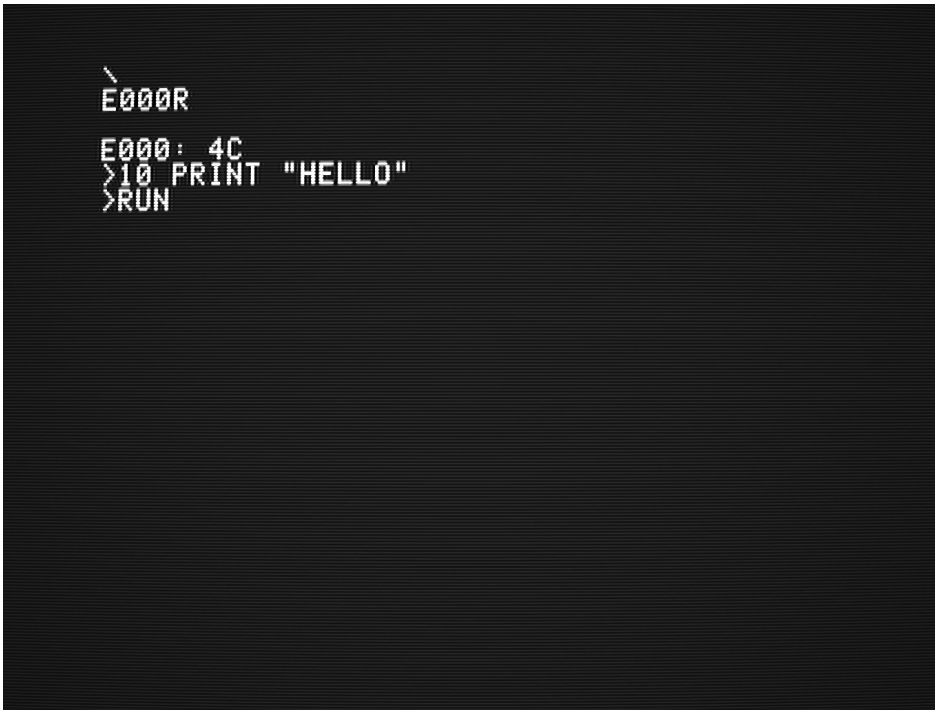


The BASIC interpreter included with this core is Integer BASIC. To start it, you have to invoke the corresponding memory address (0xE000):

```
\
E000R
```

Once the interpreter is loaded, the console changes in order to show that it is ready to accept BASIC commands:

```
\
E000R

E000:4C
>@
```



# Apple II

Based on Stephen A. Edwards Apple2fpga and vlait port for Papilio boards.

Some of its features are:

- Joystick Support (up to two fire butons)
- RAM expansion cards. 128K Saturn RAM (slot 5) + 16K Language card (slot 0).
- VGA Scanlines
- Can change monitor between colour and monochrome

For more info check ZX-Uno forum.

## SD card formatting

The SD card needs an exclusive format, so it cannot be used with other cores. It's based on concatenating `NIB` disk image file data.

To convert disk images from other format (`DSK` or `DO`), you can use `dsk2nib` utility, available in ZX-Uno SVN repository (User `guest`, password `zxuno`) and GitHub.

This process can't be undone, and it will remove any content that there was previously in the SD card.

### Windows

Concatenate the disk images (with a maximum of 20) using `COPY`:

```
COPY /B image1.nib + image2.nib + (...) + image20.nib apple2_20discs.img
```

Dump the new file to the SD card, for example, using HDD Raw Copy Tool.

### macOS and Linux

Concatenate the disk images (with a maximum of 20) using `cat`:

```
cat imagen.nib image2.nib (...) image20.nib > apple2_20discs.img
```

Dump the new file to the SD card, using `dd`:

```
sudo umount /dev/...
sudo dd if=apple2_20discs.img of=/dev/...
```

## Keyboard

### Special keys and buttons

Some versions of this core do not initialize properly the keyboard when using only a membrane keyboard (+UNO). You have to cold boot the +UNO with a PS/2 keyboard attached.

While the core is running:

- `/` (numeric keyboard): Enable or disable scanlines on VGA
- `*` (numeric keyboard): Change between colour monitor and black and white
- `F1` to `F10`: Insert disk image between 1 and 10 from the SD. Press `F12` afterwards.
- `Shift+F1` to `Shift+F10`: : Insert disk image between 11 and 20 from the SD. Press `F12` afterwards.
- `Ctrl+Alt+Backspace` (`Caps Shift+Symbol Shift+B` on +UNO): Hard reset. Backspace is the delete key, located in the top-right portion of the keyboard, above `Enter`
- `F12` (`Caps Shift+Symbol Shift+W` on +UNO): Soft reset.

# Arcades

Originally made for Zx-Uno Jamma Addon (to connect inside an Arcade Machine), there are several Arcade Game cores. Afterwards, adapted versions compatible with joysticks (like ZXUNO+ VGA+DB9 addon) were made.

There are three types:

- Vertical (the original machine used a monitor rotated 90º)
- Inverted Verticale (the machine used a monitor rotated 270º)
- Horizontal (the original machine used a horizontal monitor)

You can obtain download links for the different versions and more detailed information at ZX-Uno forum.

Take note that there is a vertical BIOS version, and a special Spectrum ROM to select and load the different Arcade Cores.

## Keyboard

### Special keys and buttons

Most of the cores have the same control keys and buttons.

Special keys that can be used while running the core:

- `1` and `2`: Player 1 and Player 2 Buttons
- `3` and `4`: Insert Coin
- Cursor keys (or joystick stick): Joystick
- `Z` and `X` (or joystick fire buttons): Fire Buttons
- `0`: When in a vertical core, enable or disable 90º rotation of directional controls
- `Scroll Lock`: Switches between composite and VGA video modes.
- `F10` (`Caps Shift+Symbol Shift+0` on +UNO): Soft Reset
- `Ctrl+Alt+Backspace` (`Caps Shift+Symbol Shift+B` on +UNO): Hard reset. Backspace is the delete key, located in the top-right portion of the keyboard, above `Enter`.

# Atari 800XL

Atari 800XL was a personal computer made by Atari in the eighties.

This core has this features:

- 320K expanded memory
- Drive support through SD
- Cart support
- Composite video and VGA
- Scanlines (VGA mode)
- Atari joystick support

## SD card

You need a SD card with the first partition in FAT32 format. A directory named `atari800` with two subdirectories: `rom` with ROMs to use (e.g: `ATARIXL.ROM`), and `user` with cart, disk files, etc. (e.g.: `ManicMin.xex`)

See the corresponding section for instructions of how to install the Atari 800XL core in ZXUNO+.

## Keyboard

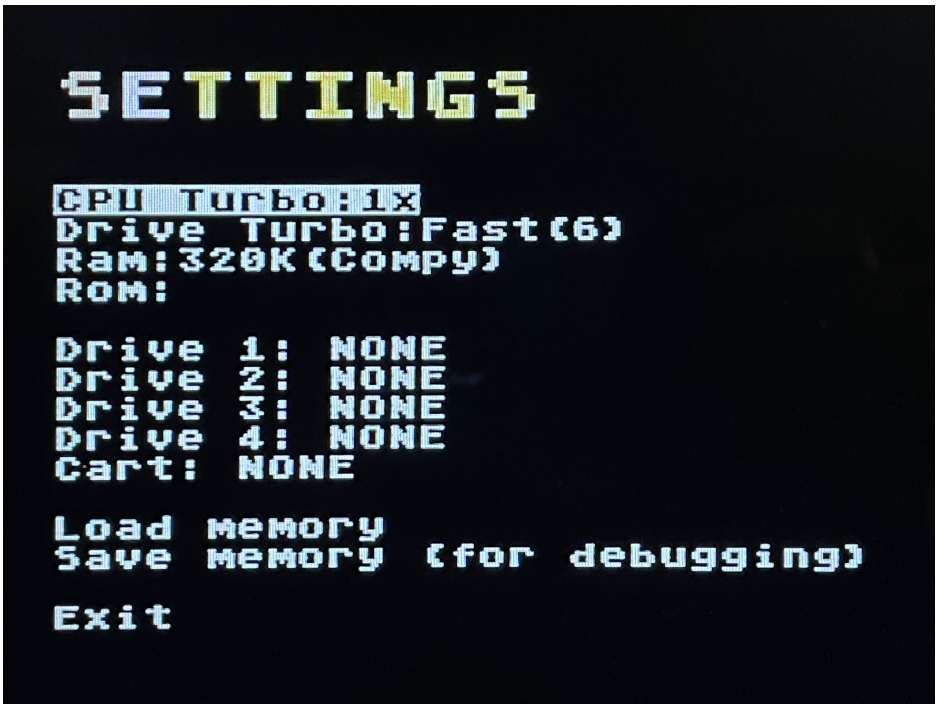
### Special keys and buttons

While the core is running:

- `Scroll Lock`: change between RGB and VGA video mode
- `-`: Enable or disable scanlines in VGA mode
- `*`: Change machine type between PAL and NTSC
- `Ctrl+Alt+Backspace` (`Caps Shift+Symbol Shift+B` on +UNO): Hard reset.
- `F5` (`Caps Shift+Symbol Shift+5` on +UNO): `Help`
- `F6` (`Caps Shift+Symbol Shift+6` on +UNO): `Start`
- `F7` (`Caps Shift+Symbol Shift+7` on +UNO): `Select`
- `F9` (`Caps Shift+Symbol Shift+9` on +UNO): `Reset`
- `F11` (`Caps Shift+Symbol Shift+Q` on +UNO): Load a disk
- `F12` (`Caps Shift+Symbol Shift+W` on +UNO): Show or hide the configuration menu
- The numeric keypad emulates a joystic, where `5` and `2` keys work as *down* direction and `0` is the fire button

## Basic Guide

Pressing F12 (Caps Shift+Symbol Shift+W on +UNO) shows or hides the configuration menu. Cursor keys and Enter (or joystick and fire button) to select and choose menu options.




The following options are available:

- CPU Turbo
- Drive Turbo
- Ram
- Rom
- Drive 1
- Drive 2
- Drive 3
- Drive 4
- Cart
- Load memory
- Save memory
- Exit

# Atari 2600

Atari 2600 is a home video game console originally branded as the Atari Video Computer System (Atari VCS).

ZXUNO+ core version is developed by Quest and DistWave.

Some of the features of the core are:

- RGB and VGA support
- Support for joysticks, keyboard, mouse and rotary encoder controls (see here for more information)

## SD card format

You need a SD card with the first partition in FAT16 format to store ROM image files of games to load.

See the corresponding section for instructions of how to install the Atari 2600 core in ZXUNO+.

## Keyboard

### Special keys and buttons

During the core execution:

- `W`, `A`, `S`, `D` or joystick 1: Directional controls for player 1
- `F` or joystick 1 fire button: Player 1 fire button
- `I`, `J`, `K`, `L` or joystick 2: Directional controls for player 2
- `H` or joystick 2 fire button: Player 2 fire button
- `Scroll Lock`: change between RGB and VGA video mode
- `Ctrl+Alt+Backspace` (`Caps Shift+Symbol Shift+B` on +UNO): Hard reset.

### Control customization

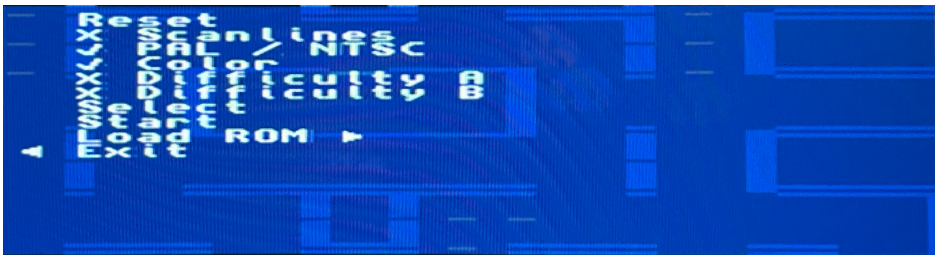
You can customize the core default controls, creating one or two files in the root of the SD card. The file has to be named `KEYSP1` for player 1 and `KEYSP2` for player 2 (you can obtain a sample file from ZX-Uno forums).

These files can be made with an hex editor (like HxD for Windows) and contain a byte sequence with this order: `Up`, `Down`, `Left`, `Right`, `Fire Button`. The codes to use are available at the Scan Codes table at the end of this manual (MAKE column), taking into account that if the code is paired with `0xE0`, the other value has to be used, adding `0x80` (e.g. for `E0 14`, it would be `0x94`).

For example, the sequence `15 1C 44 4D 29` would match `Q A O P Space`

## Basic Guide

Pressing `Esc` or joystick button 2 shows or hides the configuration menu. Cursor keys and `Enter` to select and choose menu options.




The following options are available:

- Reset core
- Scanlines
- RGB Mode (PAL/NTSC)
- Color
- Difficulty A
- Difficulty B
- Select
- Start
- Load ROM
- Exit

# BBC Micro

The BBC Micro, was a series of microcomputers and associated peripherals designed and built by the Acorn Computer company in the 1980s for the BBC Computer Literacy Project, operated by the British Broadcasting Corporation.

The ZX-Uno core has been made by Quest and enhanced after by azesmbog and hoglet.

Some of its features are:

- RGB and VGA (with optional scanlines) video output
- SD/MMC support, using `.MMB` files
- PS/2 keyboard
- Joystick support on port 1 (Emulating an analog joy)
- sn76489 sound chip implementation from PACE project (Programmable Arcade Circuit Emulation)

## SD card format

An SD card with the first partition in FAT16 or FAT32 format is needed to load software. A special ROM inside the core (MMFS in the more recent core versions, Smart SPI on older ones) which reads a `BEEB.MMB` file with disk images inside.

The file must have the name `BEEB.MMB` and it has to be in the root directory. You can make one with `MMBImager` for Windows, available at ZX-Uno SVN Repository (Usuario `guest`, contraseña `zxuno`) or with MMB/SSD Utils in perl, available at Github here or here.

The file has to be whole across the SD (not fragmented). You can use a program that can defrag files or FAT filesystems (like Defraggler for Windows) or use the following method:

1. Format the first SD partition using FAT16 or FAT32, but **NOT with quick format** (when using Windos, uncheck that option).
2. Copy `BEEM.MB` making sure it is the **FIRST file** being copied.
3. If you want you can add any other file to the SD (e.g. for using with other cores), but keep **ALWAYS** `BEEB.MMB` as the first file copied to the card.

Since the `MMB` file has the same name as the one for Acorn Electron core, you can use another core with full access to the SD card (like a a ZX Spectrum core with esxdos), with both `MMB` files with a different name, and then rename the one for BBC Micro as `BEEB.MMB` before running this core

## Keyboard


### Special keys and buttons

While the core is running:

- `Scroll Lock`: change between RGB and VGA video mode
- `-` (numeric keyboard): Enable or disable scanlines in VGA mode
- `F12` (`Caps Shift+Symbol Shift+W` on +UNO): Reset
- `Shift+F12`: Soft Reset trying to load automatically the selected disk at `BEEB.MMB`
- `Ctrl+Alt+Backspace` (`Caps Shift+Symbol Shift+B` on +UNO): Hard reset. Backspace is the delete key, located in the top-right portion of the keyboard, above `Enter`.

## Basic guide


Once the SD card is inserted and the core running, if `BEEB.MMB` file was created correctly, on start, you should see:

```
BBC Computer 32k
Model B MMFS
BASIC
>
```

Or, for a core with Smart SPI

```
BBC Computer 32k
Smart SPI
BASIC
```

Disc 0 from the image file is mounted automatically, and you can see it's contents using the command:

```
*CAT
```

To load, for example, a file name `MENU`, use the command:

```
*MENU
```

To show the list of available discs inside `BEEB.MMB` file:

```
*DCAT
```

To load a particular virtual disc:

```
*DIN discnum
```

Remember that, after inserting a disc, if it has automatic boot, you can start it pressing `Shift+F12`.

# Computers Lynx

The Lynx was an 8-bit British home computer that was first released in early 1983 as a 48kB model.Several models were available with 48kB, 96kB or 128 kB RAM.

The ZX-Uno core has been developed by Kyp069 and has these features:

- 48kB and 96 kB modes
- Optional Scorpion ROM
- Load from a external audio device
- Joystick support
- Only RGB/Composite Video out

## SD card format

This core does not use the SD card

## Keyboard

### Special keys and buttons

While running the core:

- `F6` (`Caps Shift+Symbol Shift+6` on +UNO): Switch between 48kB mode and 96kB mode (default)
- `F7` (`Caps Shift+Symbol Shift+7` on +UNO): Enable or disable Scorpio ROM
- `F8` (`Caps Shift+Symbol Shift+8` on +UNO): Switch the option to consider port $80 bits 2 and 3, so that Level 9 games are displayed properly.
- `Ctrl+Alt+Del` (`Caps Shift+Symbol Shift+N` on +UNO): Reset.
- `Ctrl+Alt+Backspace` (`Caps Shift+Symbol Shift+B` on +UNO): Hard reset. Backspace is the delete key, located in the top-right portion of the keyboard, above `Enter`.

## Basic Guide


From within BASIC, you can load from a external tape (or other external audio device) with commands like:

```
TAPE n
LOAD "NAME"
```

Where `n` is a number (between 1 and 5), and `NAME` is mandatory, and the name of the program to load.

If you don't know the name to load, you can guess with the same command sequence, but writing `LOAD ""`.

Binary files are loaded with `MLOAD` instead of `LOAD`.

Maxduino, which is used in miniduino does not, at this moment, native support for Lynx tape files.

You can use programs like Lynx2Wav with Lynx `TAP` files. The resulting audio files can be embedded inside of TSX or TZX with tools like MakeTSX or RetroConverter.

The lince script makes all this process easier, creating directly Maxduino `TZX` compatible files from Lynx `TAP` files.

# CHIP-8

CHIP-8 is an interpreted programming language, developed by Joseph Weisbecker. It was initially used on the COSMAC VIP and Telmac 1800 8-bit microcomputers in the mid-1970s. Erik Bryntse later created another interpreter based on CHIP-8, called SCHIP, S-CHIP or Super-Chip which extended the CHIP-8.

The ZXUNO+ core is based on an existing FPGA implementation of the SuperChip.

There are several sites like CHIP-8 Archive ot Matthew Mikolay's CHIP-8 where you can obtain osftware for these machines.

## SD card format

You can use a SD card with the first partition in FAT16 or FAT32 formata to store `BIN` or `CH8` ROM files to load with the core.

## Keyboard

The CHIP-8 machine uses an hexadecimal keyboard as input. This is the key mapping:

| Chip-8 | PS/2 |
|---|---|
| `1 2 3 C` | `1 2 3 4` |
| `4 5 6 D` | `Q W E R` |
| `7 8 9 E` | `A S D F` |
| `A 0 B F` | `Z X C V` |

### Special keys and buttons

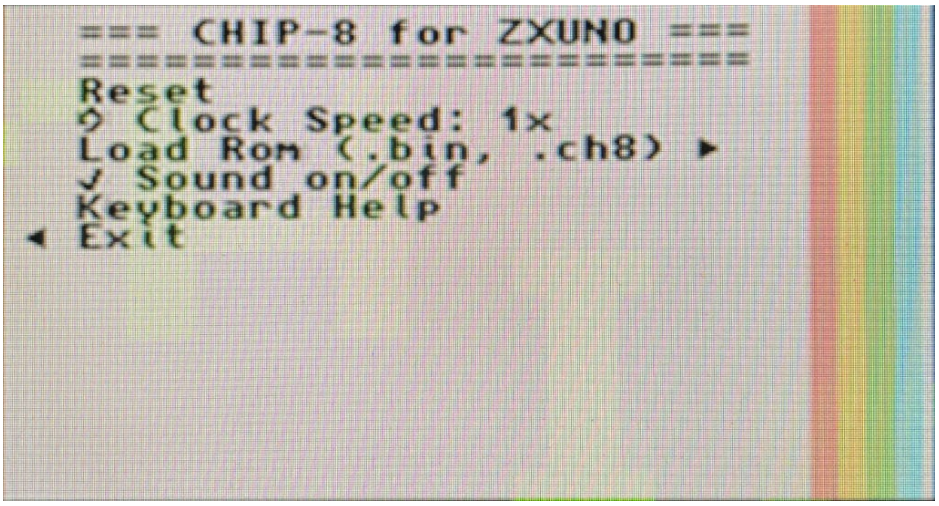
Some versions of this core do not initialize properly the keyboard when using only a membrane keyboard (+UNO). You have to cold boot the +UNO with a PS/2 keyboard attached.

While the core is running:

- `Esc`: Show or hide configuration menu
- `F11` (`Caps Shift+Symbol Shift+Q` on +UNO): Hard Reset
- `F12` (`Caps Shift+Symbol Shift+W` on +UNO): Reset

## Basic Guide

Pressing `Esc` shows or hides the configuration menu. Use the cursor keys and `Enter` to select and choose menu options.




The following options are available:

- Reset the core
- Change the core clock speed
- Load a ROM file from the SD card
- Enable or disable sound output
- Help
- Exit the menu

# ColecoVision

ColecoVision is Coleco Industries' home video-game console that was released in August 1982.

ZXUNO+ core is based on Fabio Belavenuto's project.

Some features of this core are:

- BIOS ROM is loaded from SD card
- Supports multicart ROM, also loaded from SD
- Only works with VGA

## SD card format

You need a SD card with the first partition in FAT16 format to store ROM image files of the games to load and other needed files. These can be downloaded from the original project in GitHub.

See the corresponding section for instructions of how to install the ColecoVision core in ZXUNO+.

## Keyboard

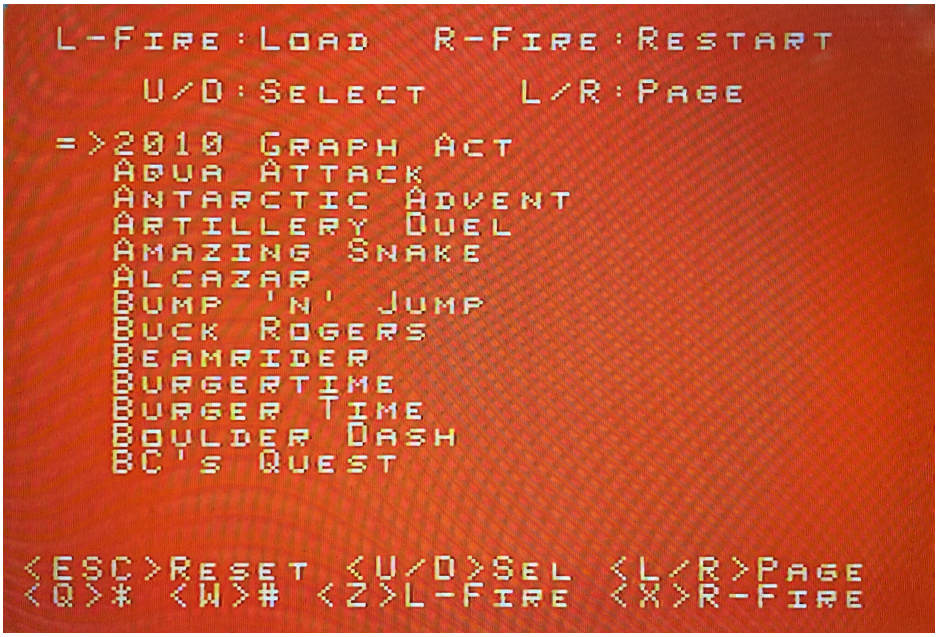
### Special keys and buttons

While the core is running:

- Cursor or `Q`, `A`, `E`, `R` or joystick 1: Directional controls for player 1
- `Z` or joystick 1 main fire button: Fire Button 1 for player 1
- `U`, `J`, `O`, `P` or joystick 2: Directional controls for player 2
- `M` or joystick 2 main fire button: Fire button 1 for player 2
- `X` or joystick 1 secondary fire button: Fire button 1 for player 1 and player 2
- `0` to `9`: Button 0 to 9 for player 1 and player 2
- `T`: Button '*'
- `Y`: Button '#'
- 'Esc': Soft Reset

## Basic Guide

On startup, BIOS ROM is loaded from the card, and then the multicart ROM.




At multicart menu, use the directional controls to choose one ROM, and then fire button 1 to load. Pressing 'Esc' restarts the core and loads the ROM selection menu again.

# Colour Genie

The EG2000 Colour Genie was a computer produced by Hong Kong-based manufacturer EACA and introduced in Germany in August 1982.

The ZX-Uno version has been developed by Kyp069 and has these features:

- 32K RAM
- Text and graphics modes
- RGB / Composite video output
- Tape loading using the audio input

## SD card format

This core does not use the SD card.

## Keyboard

The `Clear` key is not implemented.

### Special keys and buttons

While the core is running:

- `Windows`: `MODSEL`
- `F5` (`Caps Shift+Symbol Shift+5` on +UNO): NMI
- `F11` (`Caps Shift+Symbol Shift+Q` on +UNO): Hard Reset
- `F12` o `Ctrl+Alt+Del` (`Caps Shift+Symbol Shift+N` on +UNO): Reset

## Basic Guide

On boot, a question is asked about the memory size. Usually, after pressing `Enter` the BASIC interpreter is started.


From BASIC, you can load software from a external tape (or other external audio device) with the following commands.

Use `CLOAD` and `Enter` to load BASIC programs. Once the load has finished, use the `RUN` command to start.

To load machine code, type `SYSTEM` and then `Enter`. Afterwards, type the first letter of the program name and `Enter` again. When loading finishes, if the program does not start automatically, type `/` and `Enter`.

While a program is loading one `and two*` are alternately displayed on the upper right corner of the screen.

# Commodore 16

The Commodore 16 was a home computer made by Commodore International, released in 1984 and intended to be an entry-level computer to replace the Commodre VIC-20.

The ZXUNO+ core is based on FPGATED project from István Hegedus, with some changes and upgrades, like loading tapes from audio sources.

Features:

- Commodore 16 PAL expanded to 64K RAM
- 1541 Floppy, redirected to SD RAW. Read only (.D64 image files)
- Chip TED from FPGATED
- Joystick support, DB9 connector and numeric keyboard emulation
- VGA 50Hz and RGB-Composite
- VGA Scanlines
- Tape loading using the audio input
- Audio input signal polarity change support
- ROM Kernal PAL -5 modded to avoid the key press between header loading while loading from tape
- Audio out mix including audio tape feedback and TED audio
- LED used as 1541 reading activity and audio input polarity status

## SD card format

The SD card needs an exclusive format, which cannot be used with other cores. It's based on D64 concatenated images, insed 256K blocks. Downloead the file dummyto256.bin available at ZX-Uno official repository.

To include several PRG files inside a D64 disk image file, you can use DirMaster for Windows, making an image with FB16.PRG as the first program (more info at ZX-Uno forum).

The SD RAW format is compatible with the one used with Commodore 64 core, so you can use the same card, including disk images for both systems.

This process can't be undone, and it will remove any content that there was previously in the SD card.

### Windows

Concatenate the disk images using COPY:

```
COPY /B imagen1.d64 + dummyto256.bin + imagen2.d64 + dummyto256.bin + (...)
c16rawsd.img
```

Dump the new file to the SD card, for example, using HDD Raw Copy Tool.

### macOS and Linux

Concatenate the disk images using cat:

```
cat image1.d64 dummyto256.bin image2.d64 dummyto256.bin (...) > c16rawsd.img
```

Dump the new file to the SD card, using dd:

```
sudo umount /dev/...
sudo dd if=c16rawsd.img of=/dev/...
```

If it wasn't already, install Commodore 16 core into ZXUNO+.

## Keyboard

### Special keys and buttons

While the core is running:

- `Esc`: Esc
- `Tab`: RUN/STOP
- `Left Windows` = Commodore
- Numeric Keyboard: Emulated Joystick
- `F1` to `F3`: F1 to F3
- `Num Lock` or `F4`: HELP
- `Insert`: Select the first disk in the SD card
- `Page Up`: Select the next disk
- `Page Down`: Select the previous disk
- `Ctrl+Page Up`: Go 10 disks forward
- `Ctrl+Page Down`: Go 10 disks back
- `+`: Change audio input polarity
- `-`: Enable or disable scanlines in VGA mode
- `Scroll Lock`: change between RGB and VGA video mode
- `F11` (`Caps Shift+Symbol Shift+Q` on +UNO): Change joystick between port 0 and port 1
- `F12` (`Caps Shift+Symbol Shift+W` on +UNO): Soft Reset
- `Ctrl+Alt+Backspace` (`Caps Shift+Symbol Shift+B` on +UNO): Hard reset.

## Basic Guide


From within BASIC, you can load from a external tape (or other external audio device) with the command LOAD. One it's finished, type RUN and press Enter if needed.

Maxduino, which is used in miniduino does not, at this moment, support for Commodore 16 tape files, but you can use Comodoro to make compatible TZX files

To show the contents of the current disk, press F3 or use the command DIRECTORY. To load a file from disk use the command DLOAD"<file name>" and then, usually, RUN.

To load the first PRG file of a disk, press Shit+TAB or use the commadn DLOAD"*".

# Commodore 64

The Commodore 64, (C64, CBM 64/CBM64, C=64,C-64, VIC-641), was an [8-bit home computer introduced in January 1982 by Commodore International.

The ZXUNO+ core has been made by Quest.

Features:

- PAL Commodore 64 with 64K RAM
- 1541 Floppy, redirected to SD RAW. Read only
- Optional JiffyDOS either for the 1541 or for the Commodore 64. This speeds up loading
- SID sound chip
- Joystick support: physical (Atari, SMS, Megadrive...) and emulation with the numeric keyboard
- Switch VGA 50Hz / Composite-RGB
- VGA 50Hz and RGB-Composite
- VGA Scanlines
- Tape loading using the audio input
- Audio input signal polarity change support
- Modded kernel to avoid the key press after header loading when loading from tape
- LED used as 1541 reading activity

## SD card format

The SD card needs an exclusive format, which cannot be used with other cores. It's based on D64 concatenated images, insed 256K blocks. Downloead the file dummyto256.bin available at ZX-Uno official repository.

To include several PRG files inside a D64 disk image file, you can use DirMaster for Windows, making an image with FB64.PRG as the first program.

The SD RAW format is compatible with the one used with Commodore 64 core, so you can use the same card, including disk images for both systems.

This process can't be undone, and it will remove any content that there was previously in the SD card.

### Windows

Concatenate the disk images using COPY:

```
COPY /B image1.d64 + dummyto256.bin + image2.d64 + dummyto256.bin + (...) c64rawsd.img
```

Dump the new file to the SD card, for example, using HDD Raw Copy Tool.

### macOS and Linux

Concatenate the disk images using cat:

```
cat image1.d64 dummyto256.bin image2.d64 dummyto256.bin (...) > c64rawsd.img
```

Dump the new file to the SD card, using dd:

```
sudo umount /dev/...
sudo dd if=c64rawsd.img of=/dev/...
```

If it wasn't already, install Commodore 64 core into ZXUNO+.

## Keyboard

### Special keys and buttons

While the core is running:

- `Esc`: Start/Stop
- `Tab`: RUN/STOP
- `Alt` = Commodore
- Numeric Keyboard: Emulated Joystick
- `F1` to `F8`: F1 to F8
- `F9` (`Caps Shift+Symbol Shift+9` on +UNO): Pound symbol key
- `F10` (`Caps Shift+Symbol Shift+0` on UNO): `` symbol key
- `Insert`: Select the first disk in the SD card
- `Page Up`: Select the next disk
- `Page Down`: Select the previous disk
- `Ctrl+Page Up`: Go 10 disks forward
- `Ctrl+Page Down`: Go 10 disks back
- `+`: Change audio input polarity
- `-`: Enable or disable scanlines in VGA mode
- `Scroll Lock`: change between RGB and VGA video mode
- `End`: Change the colour palette
- `F11` (`Caps Shift+Symbol Shift+Q` on +UNO): Change joystick port behaviour
- `Ctrl+F12`: Soft Reset to JyffyDOS mode
- `Ctrl+F12`: Soft Reset to original C64 ROM mode (to load from tape)
- `Ctrl+Alt+Backspace` (`Caps Shift+Symbol Shift+B` on +UNO): Hard reset.

## Basic Guide


From within BASIC, you can load from a external tape (or other external audio device) with the command `LOAD`. One it's finished, type `RUN` and press `Enter` if needed.

In order to load from tape, you must use the C64 original ROM, which is enabled with `Ctrl+F12`.

Maxduino, which is used in miniduino does not, at this moment, support for Commodore 64 tape files, but you can use Comodoro to make compatible `TZX` files

To show the contents of the current disk, press `Shift+Esc` or use the command `LOAD "*",8,1`. Once you see READY on screen, use the command `RUN`.

If the disk has more than one program, press `F1` or use the command `LOAD "$"`. Then use the command `LIST` to see a list of the files in the disk.

Now, to load a specific file, use the command `LOAD "<name>",8` (where `<name>` is the name of the file to load). When you see `READY`, use the command `RUN`. Sometimes, this may not work, in this case, try to load with the command `LOAD "<name>",8,1`.

To be able to load from disk, you have to use the JiffyDOS C64 ROM, which is enabled using `F12`.

# Commodore PET

The Commodore PET (Personal Electronic Transactor) was a line of personal computers produced starting in 1977 by Commodore International.

The ZX-Uno version has been made by Jepalza, based on the pet2001fpga project.

Features:

- Tape loading using the audio input
- Only works on VGA

## SD card format

This core does not use the SD card

## Keyboard

### Special keys and buttons

While running the core:

- `F1` (`Caps Shift+Symbol Shift+1` on +UNO): Cancel tape loading
- `Alt`: Type using graphic mode

## Basic Guide


You can load from a external tape (or other external audio device) with the command `LOAD`. One it's finished, type `RUN` and press `Enter` if needed.

Maxduino, which is used in miniduino does not, at this moment, support for Commodore PET tape files, but you can use Comodoro to make compatible `TZX` files

You can use programs like Audiotap or Comodorp with Commodore `TAP` files.

# Commodore VIC-20

The Commodore VIC-20 (VC-20 in Germany and VIC-1001 in Japan) was an 8-bit home computer sold by Commodore Business Machines. The VIC-20 was announced in 1980, roughly three years after the Commodore PET.

The ZX-Uno version has been made by McLeod and Quest

Main features:

- 32K RAM expansion. It can be disabled for greater compatibility
- Tape loading using the audio input
- Audio input signal polarity change support
- Composite video and VGA support
- Joystick support

## SD card format

This core does not use the SD card

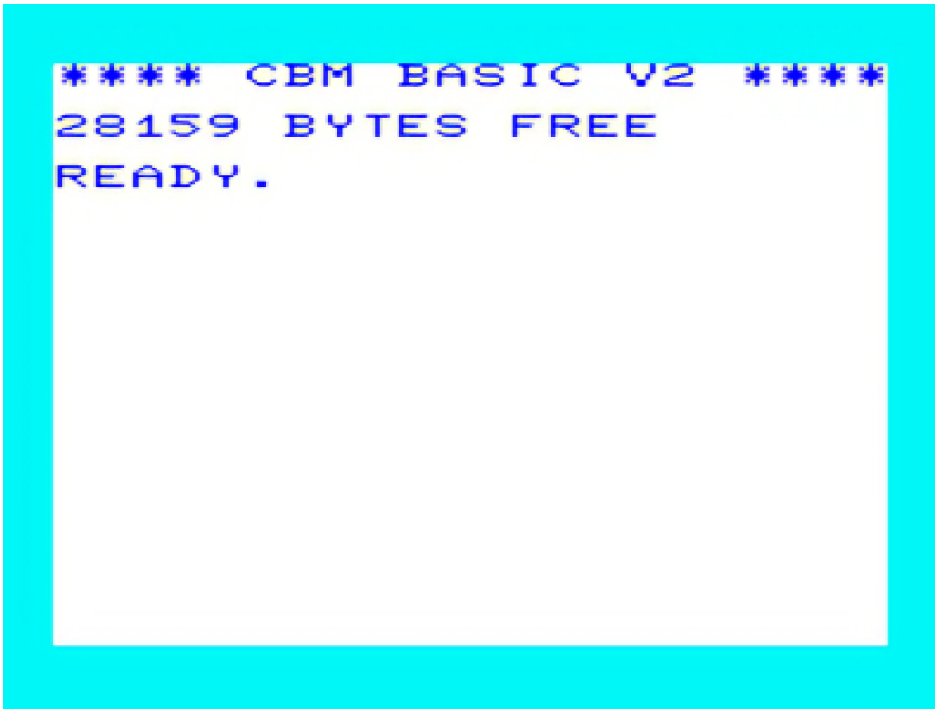
## Keyboard

### Special keys and buttons

While the core is running:

- `F9` to `F11`: Load cartridges embedded in the core
- Numeric keyboard `-`: Change audio input polarity. The red LED is enable when inverted.
- `Scroll Lock`: To switch between composite video and VGA
- `F11` (`Caps Shift+Symbol Shift+Q` on +UNO): Swtich the joystick mapping between port 0 or port 1
- `Num Lock`: Reset disabling the 32K memory expansion (enabled by default)
- `F12` (`Caps Shift+Symbol Shift+W` on +UNO): Reset enabling the memory expansion
- `Ctrl+Alt+Backspace` (`Caps Shift+Symbol Shift+B` on +UNO): Hard reset. Backspace is the delete key, located in the top-right portion of the keyboard, above `Enter`

## Basic Guide




You can load from a external tape (or other external audio device) with the command `LOAD`. One it's finished, type `RUN` and press `Enter` if needed.

Maxduino, which is used in miniduino does not, at this moment, support for Commodore VIC-20 tape files, pero es posible utilizar Comodoro para generar ficheros `TZX` compatibles

You can use programs like Audiotap or Comodoro with Commodore `TAP` files.

# Flappy Bird

Flappy Bird was very succesful mobile game. The main goal is to control a bird, attempting to fly between columns of green pipes without hitting them.

The core has been made by azesmbog using the work of Max Veerapat Kumchom, adapting the original code for the AEON Lite board to ZX-Uno.

Main features:

- Only VGA output
- It has to be controlled using a joystick

## SD card format

This core does not use the SD card

## Keyboard

This core does not use the keyboard. A joystick is needed in order to use it.

### Special keys and buttons

While the core is running:

- `Joystick Up`: Restart the game
- `Joystick Down`: Master Reset
- `Joystick Fire`: Fly

# Galaksija

Galaksija (Galaxy in english) is a build-it-yourself computer designed by Voja Antonić. It was featured in the special edition Računari u vašoj kući (Computers in your home, written by Dejan Ristanović) magazine, published 1984 in Yugoslavia.

The ZX-Uno core is a revision of the µGalaksija project, improved by azesmog, whom added the Galaksija Plus mode, and the ability to load programs from the SD card.

ZX-Uno core features:

- Only VGA output
- Sound (optional)
- Program loading from the SD card
- Galaksija Clasic (minus) y Galaksija Plus modes

## SD card format

You can use a SD card with the first partition in FAT16 format, where you can store program files, which you can get, for example at the retrospec.sgn.net web site.

## Keyboard

### Special keys and buttons

While the core is running:

- `F1` (`Caps Shift+Symbol Shift+1` on +UNO): Normal speed
- `F2` (`Caps Shift+Symbol Shift+2` on +UNO): 2x speed
- `F3` (`Caps Shift+Symbol Shift+3` on +UNO): 4x speed
- `F9` (`Caps Shift+Symbol Shift+9` on +UNO): Enable or disable sound
- `Esc`: Freezes a BASIC or machine code execution. Pressing again the same key, resumes the execution
- `Del`: Pauses a BASIC program execution while being pressed
- `End`: BREAKs a BASIC program execution
- `Shift.+M`: Enables plus mode in those programs that have the support (e.g. TETRIS.GTP)
- `M`: Gets back to normal mode from plus mode
- `F12` (`Caps Shift+Symbol Shift+W` on +UNO): Soft Reset
- `Ctrl+Alt+F5`: NMI
- `Ctrl+Alt+Del` (`Caps Shift+Symbol Shift+N` on +UNO): Soft Reset
- `Ctrl+Alt+F12`: Master Reset

## Basic guide


While in BASIC, you can enable the loading programs from the SD card with this command:

```
A=USR(&F000)
```

Then, with commands like `CD <directory>` and `DIR` you can access the directory where there is the file to load. To load a file, you have to type the name, without the extension (e.f. for a file named `GAME.GTP` you have to type `GAME`). Once the program is loaded to RAM, you can start execution with `RUN`.

The core starts up in classic Galaksija mode (minus), but you can start the Galaksija Plus mod typing:

```
A=USR(&E000)
```

You can check the available BASIC commands at Wikipedia.

# HT-1080Z

The HT-1080Z School Computer was a hungarian computer which, basically, was a Video Genie clone, which in turn is a clone of the TRS-80 Model 1 with Level II BASIC, with an added AY-3-8910 sound chip and additional input / output circuitry.

The core is based Jozsef Laszlo`s original work for MiST, ported by azesmbog to the AEON development board, and now to ZX-Uno.

It's main features are:

- New 14K BASIC ROM, which can use lowercase with the Radio Shack Lower Case Kit
- Extra FF port, with a beeper
- Included SDOS v8.4 by PVV, to load `CAS` files from the SD card
- Composite video/RGB and VGA video output

## SD card format

You can use a SD card with the first partition in FAT16 format, where you can store `.CAS` files, but with some limitations:

- Only machine code `CAS` programs. BASIC programs aren´t supported
- In order to use `CMD` programs, they have to be converted previously to `CAS` with some utility like `trld`, `trs80-tool` or `cmd2cas`
- SDOS only understands `CAS` binary format files (beysikovsky format is not supported)

You can obtain file conversion utilities at this sites:

https://www.trs-80.com/wordpress/conversion-tape-utilities/

http://48k.ca/trld.html

https://github.com/pullmoll/cass80

## Keyboard

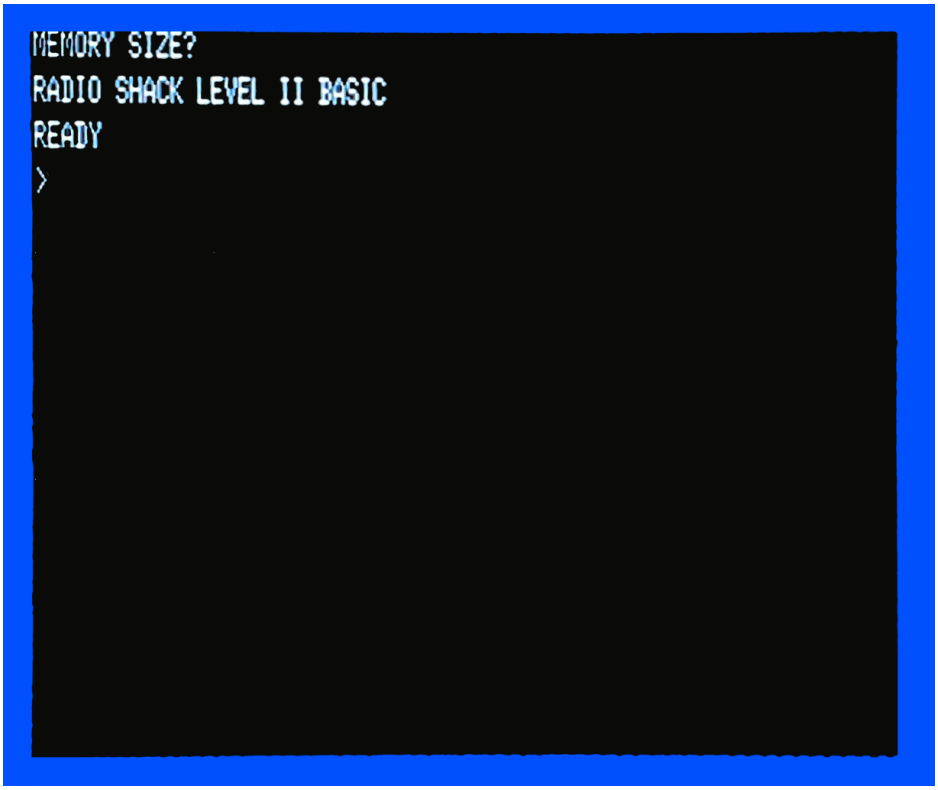
### Special keys and buttons

While the core is running:

- `Ctrl+Alt+Backspace` : Master reset
- `Ctrl+Alt+Del` : Soft reset
- `F5` : Change ink colour
- `F6` : Change paper colour
- `F7` : Change border colour
- `F9` : Select 64 or 32 columns mode (only with VGA output)
- `F10` (`Caps Shift+Symbol Shift+0` on +UNO): `PAGE` key
- `Home`: `CLEAR` key
- `Scroll Lock`: Select between VGA and PAL mode.
- `Shift+0`: If `ULCBAS.CAS` has been loadead (Upper/Lower Case Driver) changes between only uppercase mode or uppercase and lowercase mode

## Basic Guide

On startup, you are asked for the upper memory limit. Press `Enter` to continue.



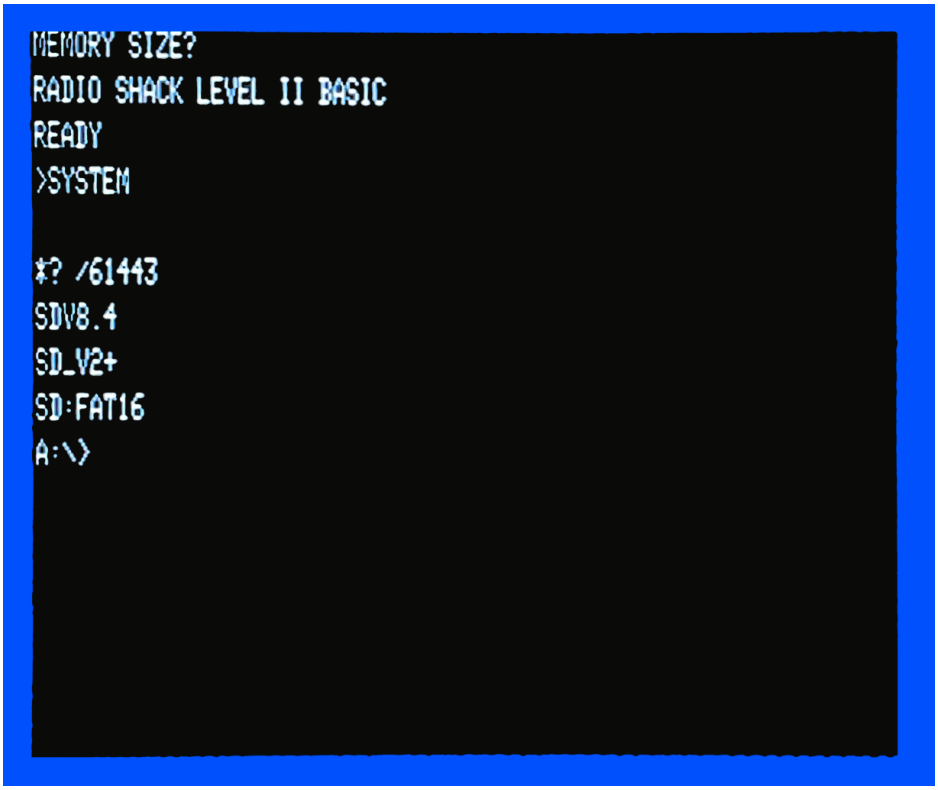

After the BASIC interpreter has loaded, if, for example, you type `PRINT MEM`, the available memory for BASIC will be displayed.

To load tape files from the SD card, you have to start SDOS typing

```
>SYSTEM
*? /61440
```

or, if you don't want the screen to be cleared

```
>SYSTEM
*? /61443
```




After that, with instructions like `CD` and `DIR` you can access where the `CAS` file to load is located. Then, to load a `.CAS` file, you only have to type the name, without extension (for example, for a file named `GAME.CAS` you have to type `GAME`).

All SDOS commands have to be typed in uppercase.

Pressing `Space` pauses the listing of files and directories, and pressing any other key afterwards, resumes it.

The included ROM has already implemented the Radio Shack Lower Case Kit. However, in order to use lowercase from BASIC, you must load from SD the Upper/Lower Case Driver (file `ULCBAS.CAS`).

# Jupiter ACE

The Jupiter ACE was a british micro computer produced by Jupiter Cantab in the early 1980s.

The ZX-Uno core has been developed by McLeod and enhanced by Azesmbog, and has, amongst others, the following features:

- Tape loading using the audio input
- Tape loading from the SD card (with SDOS)

## SD card format

You can use a SD card with the first partition in FAT16 format, where you can store Jupiter ACE `.TAP` files, which you can get, for example at the Jupiter ACE Archive website.

## Keyboard

### Special keys and buttons

While the core is running:

- `F1` to `F4`: Turbo Modes
- `F12` (`Caps Shift+Symbol Shift+W` on +UNO): Soft Reset
- `Ctrl+Alt+F5`: NMI
- `Ctrl+Alt+Del` (`Caps Shift+Symbol Shift+N` on +UNO): Reset
- `Ctrl+Alt+Backspace` (`Caps Shift+Symbol Shift+B` on +UNO): Hard reset. Backspace is the delete key, located in the top-right portion of the keyboard, above `Enter`

## Basic Guide

From within Forth, you can load from a external tape (or other external audio device) with the comand `LOAD <nombre>`.

Jupiter ACE `TAP` files are different from the ZX Spectrum ones, so, to use then with a miniduino, they have to be converted previously to `TZX` with the `acetap2zx` utility available at ZX-Uno repository (User `guest`, password `zxuno`).

To load tape files from the SD card, you have to acces first to SDOS typing

```
61440 call
```

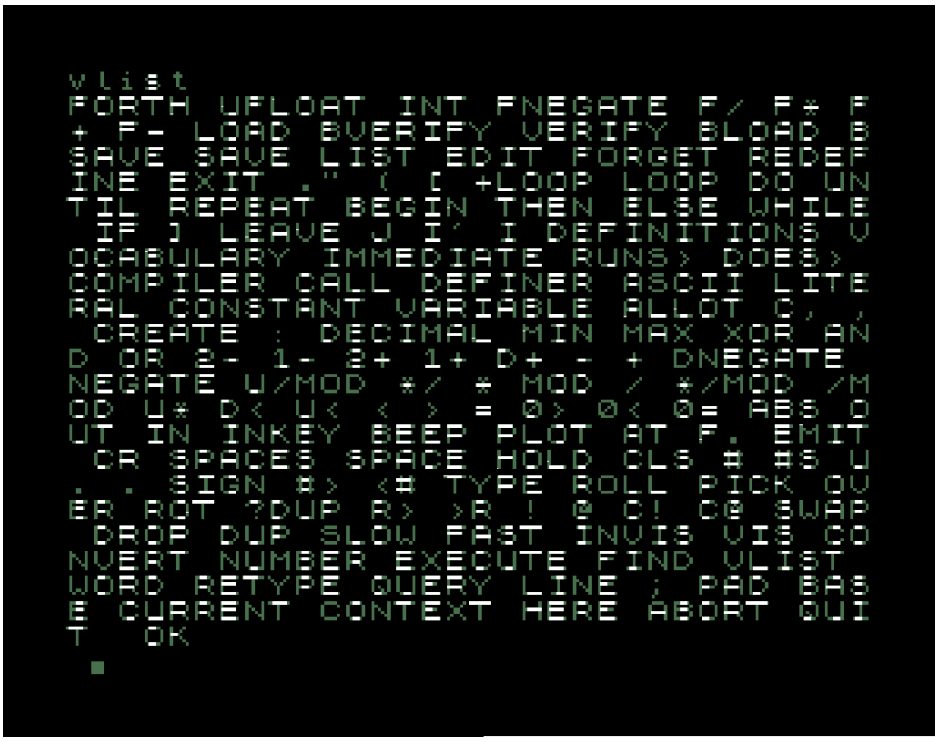
After that, with instructions like `CD` and `DIR` you can access where the `TAP` file to load is located. Then, to load a `.TAP` file, you only have to type the name, without extension (for example, for a file named `GAME.TAP` you have to type `GAME`).

All SDOS commands have to be typed in uppercase.

Once the tape has finished loading, type the corresponding command to start the program.



Instead of BASIC like other systems, Jupiter ACE uses Forth. To see the commands currently available (including those that will start a program loaded from tape, you have to type `vlist`.



# MSX

MSX1FPGA is a project to clone MSX1 in FPGA. The original development is by Fabio Belavenuto and is available at GitHub.

Some of its features are:

- MSX1 at 50Hz or 60Hz
- 128K Nextor (MSX-DOS2 evolution) ROM with SD driver
- Reconfigurable keyboard map
- Scanlines
- Joystick support

## SD card format

You have to use a SD card with the first partition in FAT16 format with code `0x06` (16-bit FAT). You can also use a second FAT16 partition for MSX software, and leaving the first one only for the system startup.

You need to get:

- Basic SD project files SD from GitHub
- Nextor driver (`NEXTOR.SYS`) and ROM (`NEXTOR.ROM`) also from GitHub
- MSX1 ROM (`MSX_INT.rom`, `MSX_JP.rom` or `MSX_USA.rom`) at the same repository

Copy the contents of the SD directory in the root of the first partition of the SD.

Copy `NEXTOR.SYS` to the same place.

Copy `NEXTOR.ROM` inside the `MSX1FPGA` directory.

Copy one MSX1 ROM (`MSX_INT.rom`, `MSX_JP.rom` or `MSX_USA.rom`) inside the `MSX1FPGA` directory, but renaming it to `MSX1BIOS.ROM`.

The file `/MSX1FPGA/config.txt` keeps the core configuration, using this format:

```
11SP01
||||||
|||||+-Scanlines: 1=Enabled, 0=Disabled
||||+--Turbo: 1=Initialize with turbo enabled
|||+---Colour System: N=NTSC, P=PAL
||+----Keymap: E=English, B=Brazilian, F=Francese, S=Spanish, J=Japanese
|+-----Scandoubler(VGA): 1=Enabled, 0=Disabled
+------Nextor: 1=Enabled, 0=Disabled
```

If it wasn't already, install MSX core into ZXUNO+.

## Keyboard

### Special keys and buttons

While running the core:

- `Print Scr`: Changes between VGA and RGB mode
- `Scroll Lock`: Enables or disables scanlines
- `Pause`: Changes between 50Hz and 60Hz
- `F11` (`Caps Shift+Symbol Shift+Q` on +UNO): Enables and disables turbo mode
- `Ctrl+Alt+Del` (`Caps Shift+Symbol Shift+N` on +UNO): Soft Reset
- `Ctrl+Alt+F12`: Hard Reset
- `Ctrl+Alt+Backspace` (`Caps Shift+Symbol Shift+B` on +UNO): Restarts the FPGA
- `Left ALT`: MSX GRAPH
- `Right ALT`: MSX CODE
- `Page Up`: MSX SELECT
- `Home`: MSX HOME (`Shift+HOME`: CLS)
- `End`: MSX STOP
- `Ñ` or `Windows`: MSX DEAD

In BASIC use `CTRL+STOP` (`Ctrl+End`) keys to stop the execution of a program.

To change the video mode between 50Hz and 60Hz (and thus play at correct speed PAL games), you can use also use `DISPLAY.COM`, which can be downloaded here.

## Basic Guide

To go to BASIC from MSX-DOS you must execute `BASIC` command.


From within BASIC, you can load from a external tape (or other external audio device) with the commands `RUN"CAS:"`, `BLOAD"CAS:",R` or `CLOAD`.

Loading from audio sources only works if turbo mode is disabled.

To go to MSX-DOS from BASIC, execute `CALL SYSTEM`.

### MSXCTRL

An exclusive utility of MSX1FPGA core, which lets you control all the core options that were previously available only by editing the configuration file or with some key combination.

When running `MSXCTRL` all the use parameters are shown:

```
MSXCTRL.COM - Utility to manipulate MSX1FPGA core.
HW ID = 06 - ZX-Uno Board
Version 1.3
Mem config = 82
Has HWDS = FALSE

Use:

MSXCTRL -h -i -r -b -[5|6] -m<0-2>
        -c<0-1> -d<0-1> -t<0-1>
        [-w<filename> | -l<filename>]
        -k<0-255> -e<0-255> -p<0-255>
        -s<0-255> -o<0-255> -a<0-255>
```

`MSXCTRL -h` show help for a parameter. For example, `MSXCTRL -i` show the current configuration, `-t 1` sets turbo mode on, etc.

### Other

There are different ways to load games depending on the kind of file: .CAS, .DSK o ROM (see this ZX-Uno forums thread for more info).

The spanish keymap officially available can be replaced with a better one. See here for more information.

# MZ-700

The MZ-700 was a personal computer sold by Sharp. It included keyboard and tape-based recorder in a single unit.

The ZX-Uno core has been made by jepalza, based on en previous implementations.

Some features of the core are:

- Only VGA video mode output
- Tape loading using the audio input
- PS/2 keyboard support trying to mimic the original keyboard layout

Since it is an experimental core, there are some problems. For example, the SRAM tends to get corrupted after some time of use. At this moment, the best option is to turn off and on the ZXUNO+ when this happens

You can use the mzf2wav utility, available for Windows, and with source code valid for Linux and Mac, to convert `MZF` tape files to `WAV` and load them with a external audio device

There is more information and software at these sites:

https://www.sharpmz.no

http://www.sharpmz.org/mz-700/monicmd.htm

http://www.sharpmz.org/mz-700/dldlang.htm

http://www.sharpmz.org/mz-700/dldgames.htm

## SD card format

This core does not use the SD card

## Keyboard

The core tries to keep the original machine keys layout.

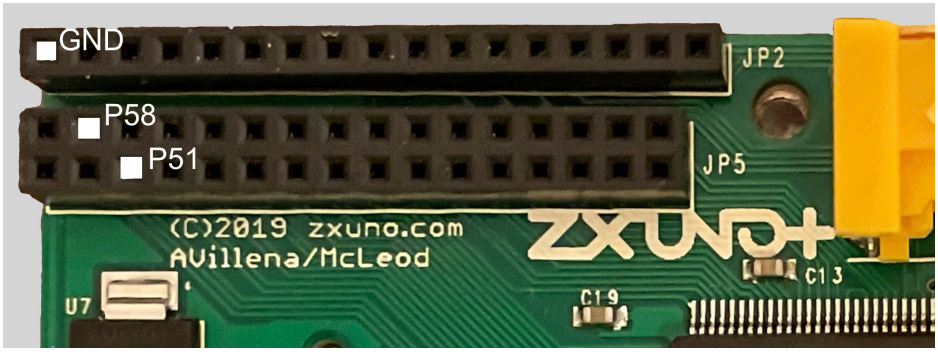
### Special keys and buttons

While the core is running:

- F1 to F5: F1 to F5
- Esc: GRAPH
- Tab: ALPHA
- Shift Lock: CTRL
- Shift: SHIFT
- Backspace: BREAK
- Enter: CR
- Insert: INS
- Del: DEL
- Cursor keys: Cursor keys

## Basic Guide

You **must** use two jump wires to expansion port pins P58 and P51, both connected to GND:




The MZ-700 had switches to determine the tape status (PLAY) and the country code. The core uses EXT9(P51) for PLAY and EXT6(P58) for the country. They are both needed since there are some games that use the japanese character set (wire to P58 **not** connected) or the european character set (wire to P58 connected), and because the tape loading routines detect the status (P51 **not connected**, Stop mode, P51 **connected**, Play mode).

As a default P58 **must** be connected to GND. You can turn it off for a japanese character set program.

Likewise, P51 **must** be connected to GND to load from tape. Some programs need to disconnect it after finishing loading.

For most software, you can leave always connected both wires.

After startup the default MZ-700 ROM has a very basic monitor with a few commands, like LOAD (L), SAVE (S), DUMP (D), etc. Afterwards, you can load from tape machine code programs or a language interpreter (BASIC, Forth, Pascal) and then load a program in the selected language.

To load a tape from the monitor with a (u external audio device):

With, at least, `P51` connected to `GND`, type `L` y `Enter` and the start the audio tape.


You can hear the audio while loading the tape. Take note, that, since the core isn't finished, there are many programs that do not work. Usually small programas load better.

# NES

Nintendo Entertainment System (also known as Nintendo NES or just NES) is the second home video game console produced by Nintendo.

The ZXUNO+ core has been made by DistWave and Quest, based on Ludde/Strigeus NES core forNexys4 board.

Some features of this core are:

- Only VGA video mode is supported, with non-accurate timings, so it may not work with some displays
- HQ2X filters that "removes pixels" from the image
- Scanlines simulation
- Made with NES NTSC clock timings, so only USA ROMs run fine. PAL ROMs run faster than they should
- You can load ROMs from the SD
- It uses all 512 KB ZXUNO+ SRAM. It's split into two 256 KB banks, uno for PRG_ROM and the other for CHR_ROM. So any ROM using more than 256 KB for CHR or PRG won't work. Obviously any ROM bigger than 512 KB won`t work too.
- Joystick support

## SD card format

You need a SD card with the first partition in FAT16 format to store ROM image files of the games to load. ROM files can be inside subdirectories.

See the corresponding section for instructions of how to install the NES core in ZXUNO+.

## Keyboard

### Special keys and buttons

While the core is running:

- `Esc` or joystick button 2: Show or hide configuration menu
- Cursor keys, and `Enter` to use the menu
- `Ctrl+Alt+Backspace` (`Caps Shift+Symbol Shift+B` on +UNO): Hard reset

There are alternate versions of this core, which can also be controlled with a keyboard:

- `W`, `A`, `S`, `D` (customizable) or joystick 1: Directional controls for player 1
- `F` and `R` (customizable) or joystick 1 fire buttons: Player 1 A and B buttons
- `5` y `1`: `Select` and `Start` (Player 1)
- `I`, `J`, `K`, `L` (customizable) or joystick 2: Directional controls for player 2

- H, Y (customizable) or joystick 2 fire buttons: Player 2 A and B button
- 6 y 2: Select and Start (Player 1)
- F4 (Caps Shift+Symbol Shift+4 on +UNO): Swap the keyboard keys for player 1 and player 2
- Ctrl+Alt+Backspace (Caps Shift+Symbol Shift+B on +UNO): Hard reset. Backspace is the delete key, located in the top-right portion of the keyboard, above Enter

### Control customization

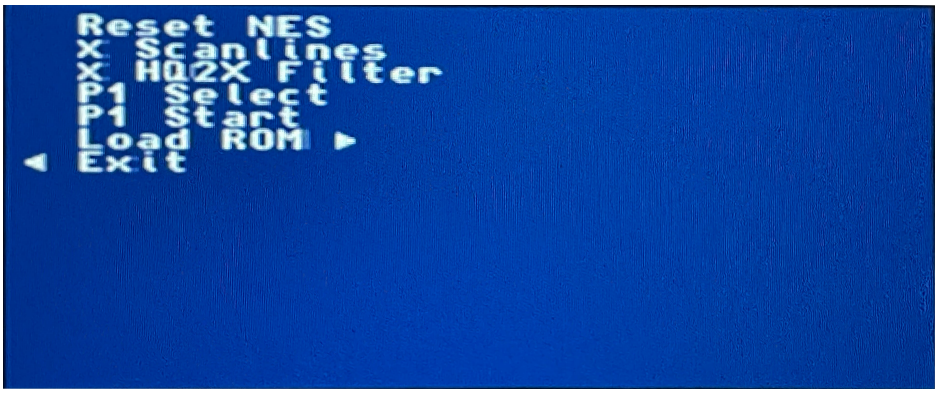
You can customize the core default controls, creating one or two files in the root of the SD card. The file has to be named KEYSP1 for player 1 and KEYSP2 for player 2 (you can obtain a sample file from ZX-Uno forums).

These files can be made with an hex editor (like HxD for Windows) and contain a byte sequence with this order: Up, Down, Left, Right, Button A, Button B. The codes to use are available at the Scan Codes table at the end of this manual (MAKE column), taking into account that if the code is paired with 0xE0, the other value has to be used, adding 0x80 (e.g. for E0 14, it would be 0x94).

For example, the sequence 15 1C 44 4D 29 22 would match Q A O P Space X, and the sequence 1D 1B 23 2B 1A 22 would be W S D F Z X

## Basic Guide

Pressing Esc or joystick button 2 shows or hides the configuration menu. To navigate the menu and activate or choose any option, use the cursor keys and Enter.




The following options are available:

- Reset NES
- Scanlines
- HQ2X Filter
- P1 Select
- P1 Start
- Load ROM
- Exit

You can move faster through the menu options using Page Up and Page Down.

# Ondra SPO 186

The Ondra SPO 186 was a Czechoslovak 8-bit computer developed in 1985 at Tesla Liberec as a cheap school and home computer.

The ZX-Uno version has been developed by PetrM1, based on the previous MiSTer version, and has these features:

- 64 kB RAM
- Keyboard support
- Sound
- Software loading via ZX-Uno audio in port
- Program loading via serial port

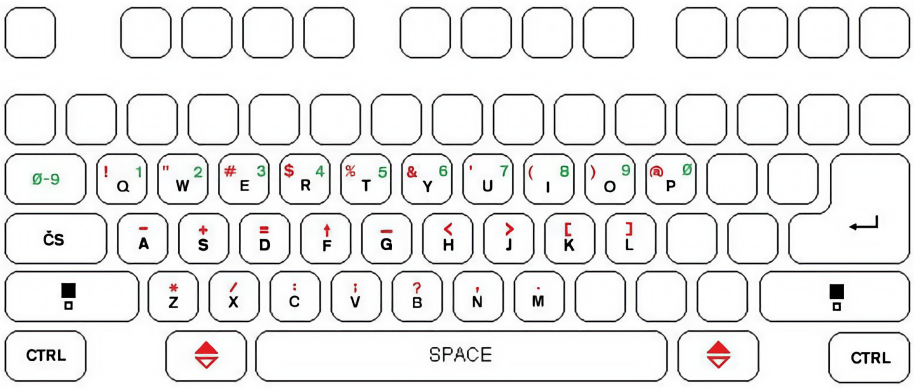
## SD card format

This core does not use the SD card.

## Keyboard

Due to the requirements of this core, it is not enough having a membrane keyboard connected. It is necessary to have a PS/2 keyboard connected at the same time to be able to use it.

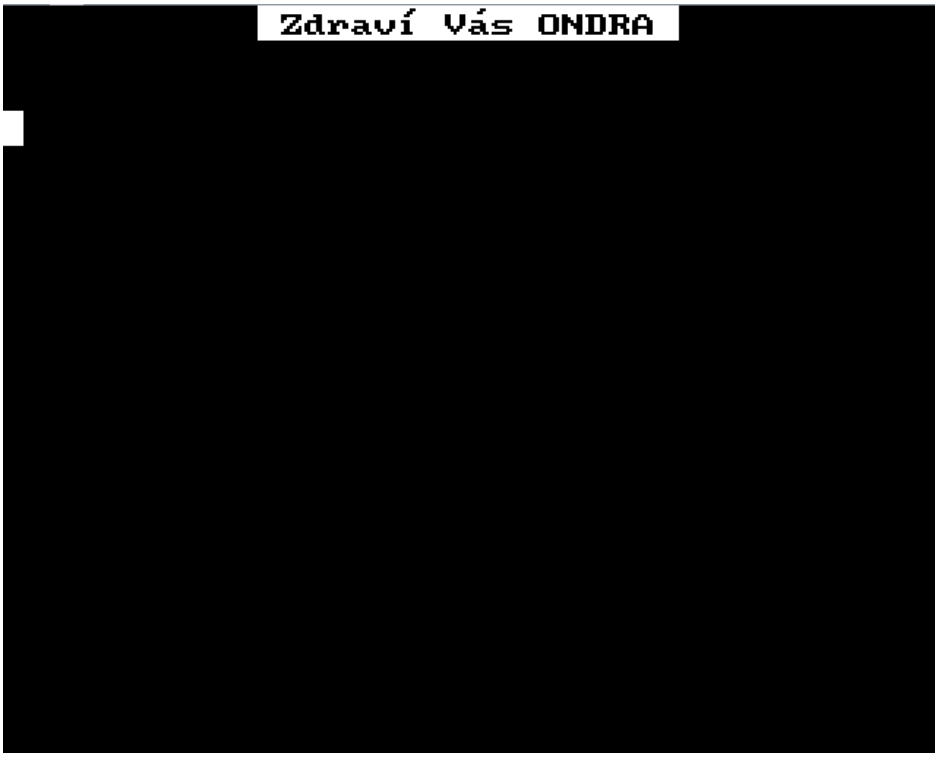
Ondra has a 37-key keyboard, where the use of modifier keys allows each key to serve 2 or 3 different characters. Each modifier key must be pressed (and held) before pressing the character key to be modified.




### Special keys and buttons

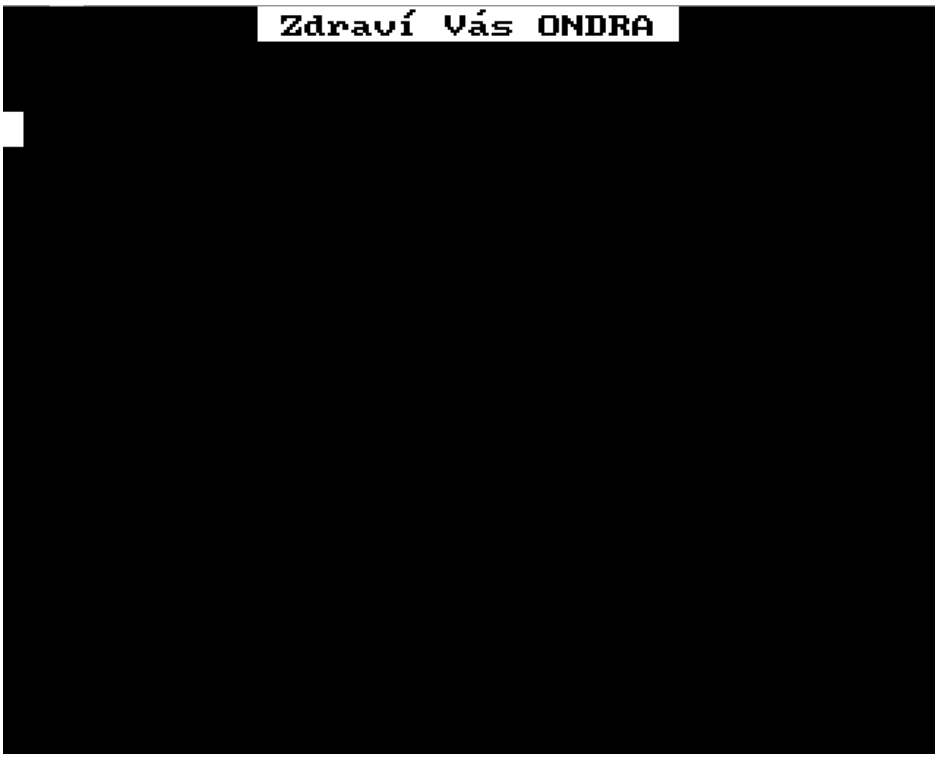
- `Shift`: `Shift`
- `Alt`: `Symbols`
- `Tab`: `Numbers`
- `Control`: `Ctrl`
- `Shift`: `ČS` (used for Czech diacritics)

## Basic guide




Once you have started the core, you can load it from a tape (or other external audio device) by pressing the `Enter` key (if you have typed anything, erase it all using `Ctrl+Left Cursor`). The display will then start to flash and the following text will be shown:

```
.KÓD 1
```




Start tape playback and the text on the screen will change during loading.

# Oric Atmos

The Oric Atmos was the name used by UK-based Tangerine Computer Systems for one 6502-based home computer sold in the 1980s, primarily in Europe.

It has been developed by Kyp069 and has these features:

- VGA / RGB video Out
- NMI
- DOS-8D (based on Pravetz 8d, a bulgarian clone of Oric Atmos)

For more information, see the ZX-Uno forum.

## SD card format

The SD card needs an exclusive format, so it cannot be used with other cores. It's based on concatenating `NIB` disk image file data. To convert disk images from other format (`DSK` or `DO`), you can use `dsk2nib` utility (the resulting image files must be 232960 bytes long).

This process can't be undone, and it will remove any content that there was previously in the SD card.

### Windows

Concatenate the disk images (with a maximum of 20) using `COPY`:

```
COPY /B image1.nib + image2.nib + (...) + image20.nib oric_discs.img
```

Dump the new file to the SD card, for example, using HDD Raw Copy Tool.

### macOS and Linux

Concatenate the disk images (with a maximum of 20) using `cat`:

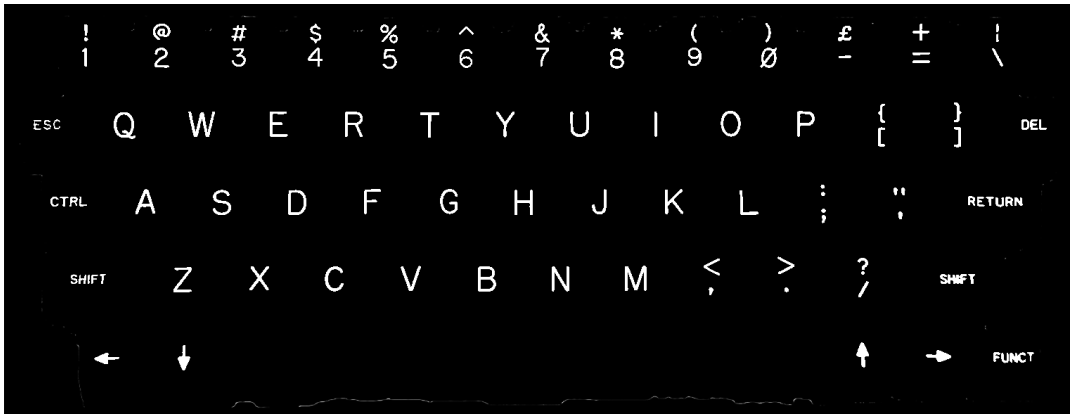
```
cat imagen.nib image2.nib (...) image20.nib > oric_discs.img
```

Dump the new file to the SD card, using `dd`:

```
sudo umount /dev/...
sudo dd if=oric_discs.img of=/dev/...
```

## Keyboard

This is the keyboard layout:

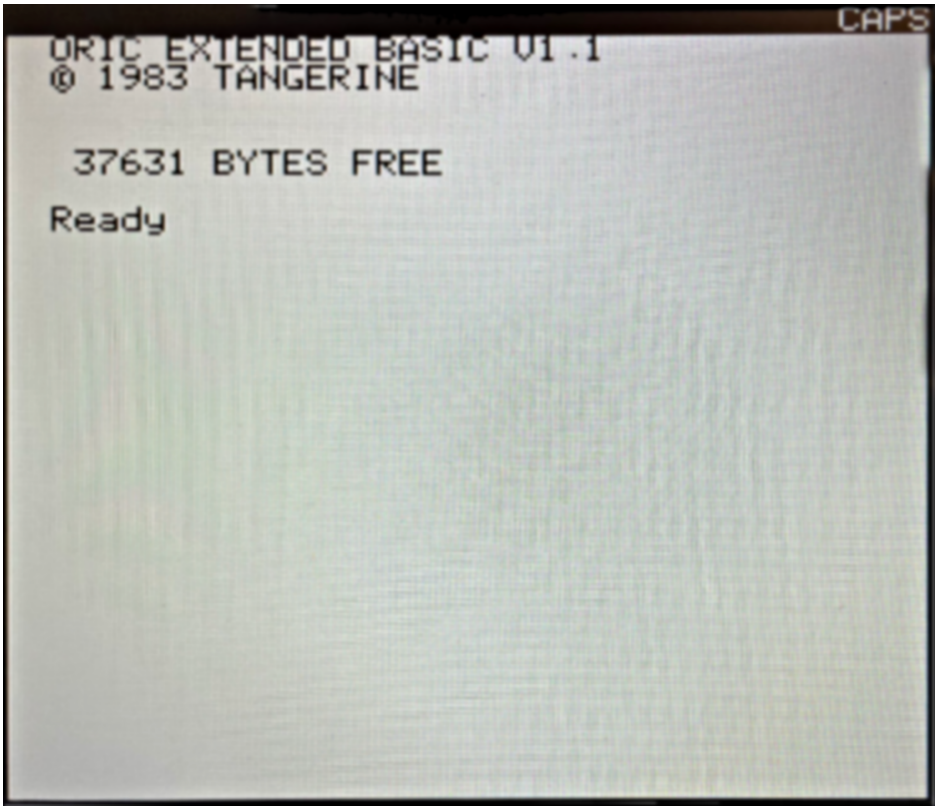



### Special keys and buttons

While the core is running:

- `Scroll Lock`: Change between RGB/Composite and VGA video mode
- `Home`: Show debugging data on screen (Current disk track -two digits-, disk image number -two digit-, and program counter -four digits-)
- `End`: NMI
- `Home+End`: Reset
- `Page Down`: Load the next disk
- `Page Up`: Load the previous disk

## Basic Guide




You can load and enable DOS-8D with the command:

```
CALL800
```

Then, you can show the files in the current disk with the `DIR` command.

Yo load and run a program, you have to type `-` and the name of the file. This way, for a program named `DEFENDER.EXE`:

```
-DEFENDER.EXE
```

# Oric Atmos (Kyp)

A different core version of Oric Atmos, based on Rampa's implementation for MiST, Mistica and SiDi.

It has been developed by Kyp069 and has these features:

- VGA (Scandoubler) and composite video out
- 64KB RAM and 16KB ROM. The ROM hides the upper memory, but it can be accessed all the memory banks using machine code
- You can choose the ROM between the original version (1.1) and version 1.22
- Joystick directional controls are mapped to the cursor keys and both fire buttons to the keys Space and S
- Program loading using the audio input
- Program saving using the audio output

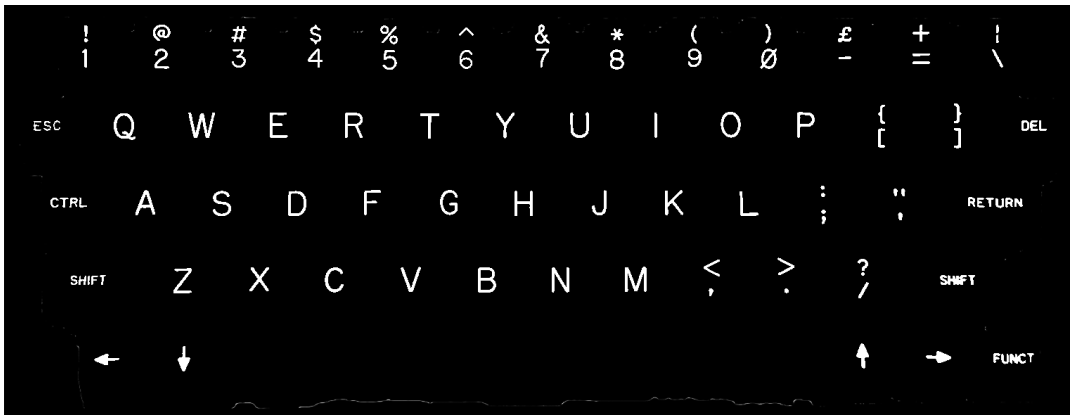
For more informatio you can see RetroWiki forums.

## SD card format

This core does not use the SD card.

## Keyboard

The keyboard layout tries to mimic as much as possible a real Oric Atmos.

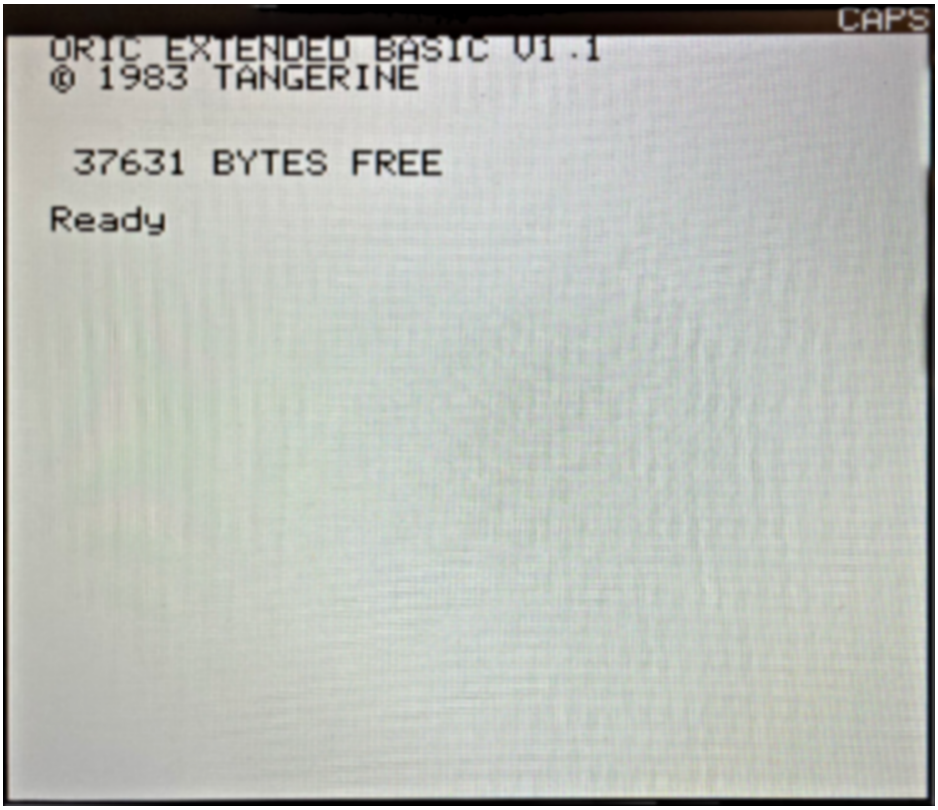


### Special keys and buttons

While the core is running:

- F1 : Choose ROM 1.1
- F2 : Choose ROM 1.22
- F5 (Caps Shift+Symbol Shift+5 on +UNO): NMI
- F10 (Caps Shift+Symbol Shift+0 on +UNO): Enable audio output when using CSAVE
- Ctrl+Alt+Backspace or F11: Hard reset. Backspace is the delete key, located in the top-right portion of the keyboard, above Enter.
- Ctrl+Alt+Del or F12: Soft reset
- Scroll Lock: Selects between VGA and composite video modes
- Alt Gr.: FUNC key
- Esc: ESC key
- Del: DEL key
- Ctrl: CTRL key
- Shift: SHIFT key
- Left Windows, Left Alt, Right Windows, Menu: Cursor keys Left, Down, Up, Right

## Basic Guide




From BASIC, you can load software from a external tape (or other external audio device) using the command `CLOAD` and `Enter`. A message will show on screen

```
Searching...
```

When a program load is detected, the text changes to

```
Loading...<program name>
```

There's a script named `calorico.sh` at RetroWiki forums, made with bash, PHP and TSXphpclass utilities from Natalia Pujol to convert Oric `.TAP` files into `.TSX`.

To save a BASIC program with the audio output, you can use the command `CSAVE "NAME"` and `Enter`. If you want to hear while recording, enable the recording audio previously using the `F10` key.

**Do not** keep recording audio enabled (`F10`) while loading using the audio input. This is because the audio feedback has a high pitch and volume sound which breaks audio loading.

# PC XT

Next186lite core is an implementation of Next186 core for FleaFPGA and Papilio Pro, but adapted and trimmed.

Features:

- Only works on VGA out
- Next186 core at 25 MHz and 50 MHz system bus. Te processor is 286 equivalent in real mode. There's no protected mode.
- 504 KB conventional RAM
- 60 KB VRAM
- PS/2 keyboard and mouse support
- 80x25 text mode
- EGA 320x200x16 and MCGA 320x200x256 graphic modes, partially functional (MCGA mode does not draw the last lines on screen because 2.5 KB video RAM is missing). No graphic card is being emulated
- Beeper Sound and Tandy 3 voice
- Parallel DAC port to play digital sound

## SD card format

You have to use a SDHC card, with the first partition in FAT16 format and with MS-DOS (or similar) installed. You can achieve this using, for example, virtualization software and connecting directly de SD card device as a hard disk. You can also find several pre-made images at ZX-Uno forum page and ZXDOS+ forums.

See the corresponding section for instructions of how to install the PC XT core in ZXUNO+.

# PC XT CGA

The IBM Personal Computer XT also known as PC/XT was the second PC IBM model. The IBM Color/Graphics Adapter was the first graphic adapter card made by IBM for PC.

This core, made by spark2k06, and based on the original port of Next186 by DistWave, tries to be a more faithful recreation of the original PC XT, and has these features:

- Faithful implementation of Motorola 6845 chip, having all its registers and graphic modes
- Same behaviour with PUS SP like a 8086/80186 the CPU is correctly detected
- All partial video modes in Next186 are removed (MCGA/EGA/VGA)
- Full CGA implementation based on Graphics Gremlin project by TubeTime
- Partial Tandy Graphics (TGA) implementation
- Simplified clocks, setting the overall speed of the machine to 12Mhz (default) or 4.77Mhz (Throttle Down)
- Revised BIOS video functions, using part of Micro8088 project by Sergey Kiselev
- RGB666 video DAC support
- 2MB SRAM version has, for the moment, 1MB (EMS) available for the core

## SD card format

You have to use a SDHC card, with the first partition in FAT16 format and with MS-DOS (or similar) installed. You can achieve this using, for example, virtualization software and connecting directly de SD card device as a hard disk. You can also find several pre-made images at ZX-Uno forum page and ZXDOS+ forums.

**Do not use any FAT reordering utility** since, this may stop DOS from booting.

Some files and directories used by MSXDOS have the same names, so it's not recommended to use the same SD card for MSX and PC XT cores.

See the corresponding section for instructions of how to install the PC XT core in Zx-Uno.

## Keyboard

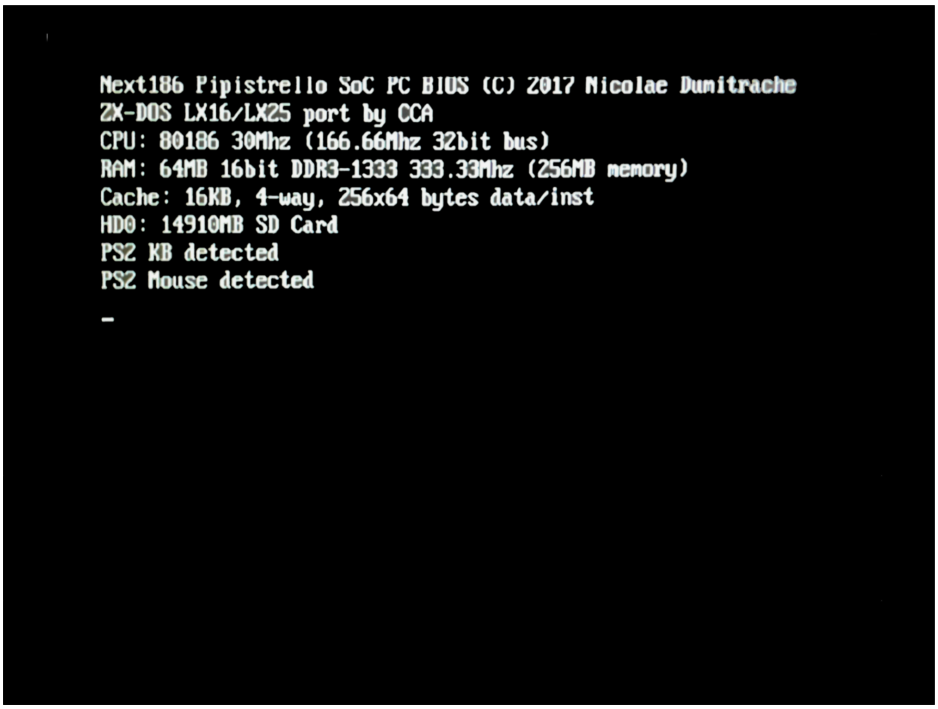
### Special keys and buttons

The following +UNO keyboard combinations are obtained while in PCXT mode (Caps Shift+Symbol Shift+U and then 9). Please read the section dedicated to +UNO keyboard modes for more information.

While the core is running:

- Ctrl+Alt+Scroll Lock: Change between Green, amber, and black and white modes
- Ctrl+Alt+KeyPad -: Throttle down speed
- Ctrl+Alt+KeyPad +: Set speed to normal (12MHz)
- Ctrl+Alt+Del (Caps Shift+Symbol Shift+N on +UNO): Soft Reset
- Ctrl+Alt+Backspace (Caps Shift+Symbol+B on +UNO): Hard reset. Backspace is the delete key, located in the top-right portion of the keyboard, above Enter

## Basic Guide




1. Start the ZX-Uno
2. For a +UNO, change the keyboard mode to PC XT keyboard mode (Caps Shift+Symbol Shift+U, and then 9)
3. Reboot the ZX-Uno without losing the temporary keyboard mode (Caps Shift+Symbol Shift+`B)
4. Quickly, press Caps Shift+2 or Caps Shift+6 and select the PC XT core to boot

On a +UNO, it's not recommended to use the PC XT keyboard mode with a PS/2 keyboard connected. This causes some conflicts and both keyboards won't function properly.

# Pong

Pong was one of the earliest arcade video games manufactured by Atari.

The latest ZX-Uno core has been made by avlixa, and some of its features are:

- Two different video modes: Composite Video/RGB and VGA
- 7 game variants
- Support for 2 or 4 players
- Support for Joysticks, keyboard, mouse and rotary encoder controls (see here for more information)
- Several colour modes

## SD format

This core does not use the SD card.

## Keyboard

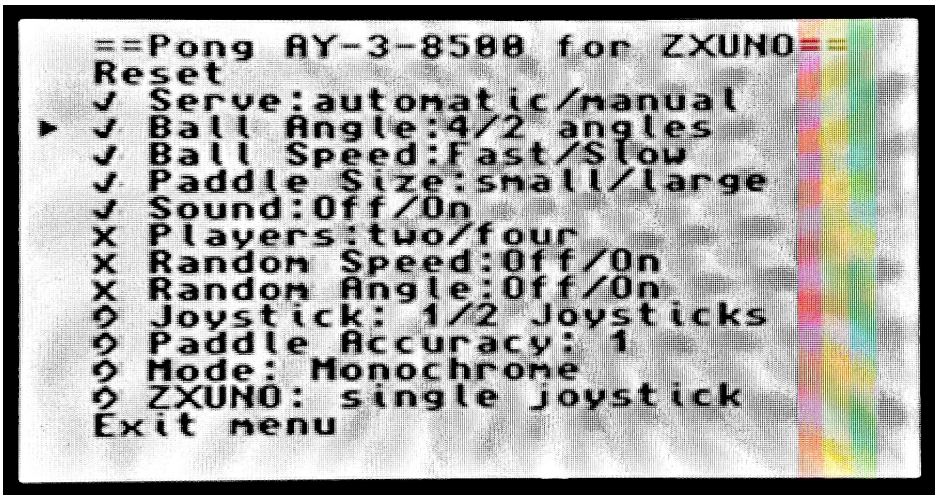
### Special keys and buttons

While the core is running:

- `Esc` or joystick button 2: Show or hide configuration menu
- `Ctrl+Alt+Backspace` (`Caps Shift+Symbol Shift+B` on +UNO): Hard reset
- `Scroll Lock`: switch between VGA and RGB mode
- `F3` o `F12`: Restart game
- Number between `1` and `7`: Change the game variant
- Joystick 2 (right): Control right pad (Player 1).
- Joystick 1 (left): Control left pad (Player 2)
- `Cursor up` and `Cursor down` or `O` and `K`: Control right pad (Player 1 in 2 player mode and player 3 in 4 player mode)
- `Q` and `A`: Control left pad (Player 2 in 2 player mode and player 4 in 4 player mode)
- `Z`, `M` or joystick button 1: Manual serve
- Cursor keys and `Enter` to use the menu

## Basic Guide

Pressing Esc or joystick button 2 shows or hides the configuration menu. Cursor keys and Enter to select and choose menu options.




The following options are available:

- Serve mode
- Ball Angle
- Ball Speed
- Paddle Size
- Sound
- Number of players
- Speed mode
- Angle mode
- Joystick, mouse, etc. controls
- Paddle accuracy
- Colour mode
- Exit

# SAM Coupé

The SAM Coupé was an 8-bit British home computer that was first released in late 1989. It was designed to have compatibility with the Sinclair ZX Spectrum, but in 48K mode only.

The ZX-Uno cores has these features:

- VGA (con scanlines) and RGB / Composite Video Out
- Full keyboard support
- SAA1099 sound
- Software loading via ZXUNO+ audio in port

## SD card format

This core does not use the SD card

## Keyboard

### Special keys and buttons

While the core is running:

- `Ctrl+Alt+F5`: NMI.
- `Ctrl+Alt+Del` (`Caps Shift+Symbol Shift+N` on +UNO): Soft reset.
- `Ctrl+Alt+Backspace` (`Caps Shift+Symbol Shift+B` on +UNO): Hard reset. Backspace is the delete key, located in the top-right portion of the keyboard, above `Enter`.

## Basic Guide


You can only load software from a external tape (or from other external audio device) with the command `LOAD""` (or pressing `7` of the numeric keyboard, which is like pressing `F7` of the SAM Coupé keyboard).

At ZX-Uno SVN repository (here - User `guest`, password `zxuno`) you can find several programs adapted to TZX tape file.

# Sega Master System

The Sega Master System was a video game console manufactured by Sega.

The ZX-Uno core is based on Ben's original work for Papilio Plus.

Some of the core features are:

- VGA and Composite or RGB video output with PAL sync (50Hz)
- Joysticks support
- Joystick emulation via keyboard
- Loading cartridge image files from the SD card

## SD card format

You need a SD card with the first partition in FAT16 format to store `BIN` or `SMD` image files of the games to load. ROM files can be inside subdirectories.

See the corresponding section for instructions of how to install the Sega Master System core in ZXUNO+.

## Keyboard

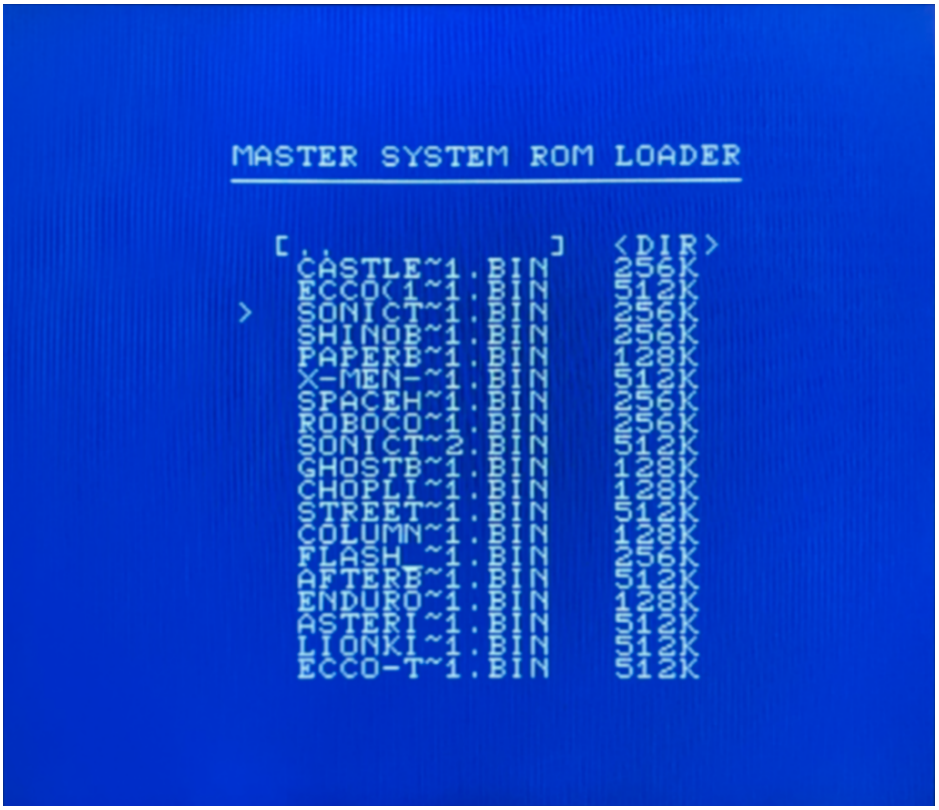
### Special keys and buttons

While the core is running:

- Cursor or `Q`, `A`, `O`, `P` or joystick 1: Directional controls for player 1
- `Z`, `Windows` or joystick 1 main fire button: Fire Button 1 for player 1
- `X`. `Space` or joystick 1 secondary fire button: Fire button 1 for player 1
- `Pause`: Master System pause button
- `Scroll Lock`: Change between RGB/Composite and VGA video mode
- `Ctrl+Alt+Backspace` (`Caps Shift+Symbol Shift+B` on +UNO): Hard reset
- `F12` (`Caps Shift+Symbol Shift+W` on +UNO): Reset and show the file selection menu

## Basic Guide

Pressing F12 shows or hides the file selector. Use the cursor keys and Enter to load the selected file.



# SmartROM

SmartROM is a kind of firmware that can load different ROMS for implementations of the ZX-Uno core in FPGA boards without flash memory, or where the flash memory cannot be used by the ZX-Uno core.

## SD card format

The card has to be formatted the same as when preparing for the main Spectrum core with, at least, one FAT16 or FAT32 partition.

Also, you have to install esxdos 0.8.8 to the card, and create a directory named `ZXUNO` with, at least, the `SMARTROM.ZX1` file and a RomPack file named `ROMS.ZX1` inside.

Finally, if your PS/2 keyboard is not in Spanish, you can copy the corresponding keyboard map file, renamed as `KEYMAP.ZX1`.

See the corresponding section for instructions of how to install the SmartROM core in ZXUNO+.

# Vectrex

The Vectrex was a vector display-based home video game console. It also had an integrated monochrome CRT monitor.

The ZX-Uno core is based on DarFPGA`s original work.

Some of its features are:

- VGA video out
- OSD Menu
- Joystick and keyboard controles
- Maximum supported ROM size: 8K

## SD card format

You need a SD card with the first partition in FAT16 or FAT32 format to store `BIN` or `VEC` files of the software to to load. Those files can be inside subdirectories.

At archive.org you can obtain software for Vectrex.

See the corresponding section for instructions of how to install the Vectrex core in ZXUNO+.

## Keyboard

### Special keys and buttons

During the core execution:

- Cursor or joystick: Directional controls
- `Z`: Fire button 1
- `X` or main joystick button: Fire button 2
- `C` or secondary joystick button: Fire button 3
- `V`: Fire button 4
- `Esc`: Show or hide the menu
- `Ctrl+Alt+Backspace` (`Caps Shift+Symbol Shift+B` on +UNO): Hard reset
- `F12` (`Caps Shift+Symbol Shift+W` on +UNO): Reset

## Basic guide

By default, Vectrex starts with the built in game (Mine Storm).


Pressing `Esc` shows or hides the configuration menu. Cursor keys and `Enter` are used to select and choose menu options.

The following options are available:

- Reset core
- Disable bluish color and enable white color for vectors
- Load ROM file from SD
- Exit the menu

# Videopac

Philips Videopac, also known as Magnavox Odyssey², Philips Videopac G7000 o Philips Odyssey², is a second generation home video game console that was released in 1978.

The ZXUNO+ core is make by avlixa, and is based on ZXDOS core by yomboprime.

Some features of the core are:

- RGB and VGA support
- Needs at least one joystick to be used
- Different colour modes including monochrome
- loadable VDC ROM charset for some custom ROMs

## SD card format

You need a SD card with the first partition in FAT16 format to store ROM image files to load.

See the corresponding section for instructions of how to install the Videopac core in ZXUNO+.

## Keyboard

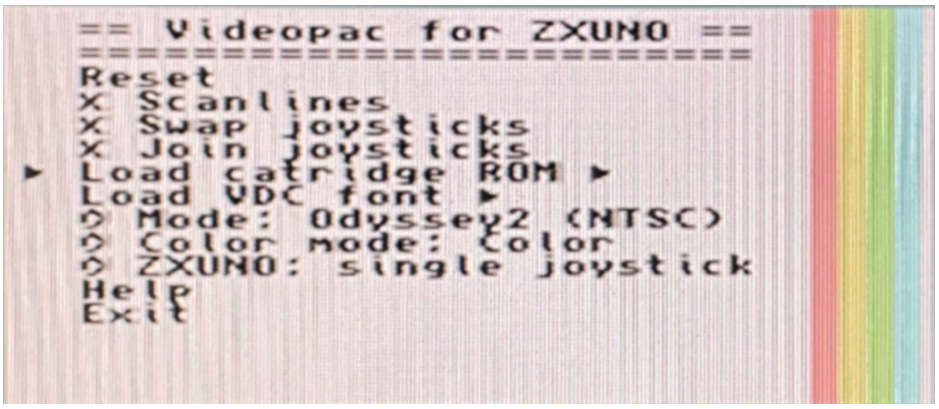
### Special keys and buttons

During the core execution:

- `Scroll Lock`: Change between RGB/Composite and VGA video mode
- `F2` (`Caps Shift+Symbol Shift+2` on +UNO): Switch between RGB and Composite when not in VGA video mode
- `F3` (`Caps Shift+Symbol Shift+3` on +UNO): Reset
- `Ctrl+Alt+Backspace` (`Caps Shift+Symbol Shift+B` on +UNO): Hard reset
- After loading a ROM, most games will prompt the user with "SELECT GAME". Press `0-9` on the keyboard or mapped controller button to play the game.
- `Esc` (`Caps Shift+Space` on +UNO) or joystick button 2 to show or hide the options menu
- `W`, `A`, `S`, `D` or cursor keys and then `Enter` to choose and select menu options

## Basic Guide

Pressing `Esc` (`Caps Shift+Space` on +UNO) or joystick button 2 shows or hides the configuration menu. Cursor keys and `Enter` to select and choose menu options.




The following options are available:

- Reset core
- Scanlines
- Swap Joysticks
- Join Joysticks
- Load Cartridge ROM
- Load VDC Font
- Video mode: PAL/Videopac or NTSC/Odyssey2
- Color Mode
- Exit

Note also that the system did not have a well defined player 1 or player 2 controller, and some games may alternate on a game-to-game basis. You may need to swap controllers to use the input or (for one player games) use the join joystick option of the menu

Usually, there is no on-screen display of the game options, so looking at the instruction manuals (for example following this link) may be helpful in selecting a game.

If, when browsing the ROM directory, you can't see all of them, try to split the content into several subdirectories with less files per directory.

### Change VDC ROM charset

You can, for some ROMs, load a `CHR` file including a custom font, instead of the original one which was included with the Intel 8244/8245 chip.

Those files can be made following the instructions and using the editor available at the project repository, following this link.

# ZX81

The ZX81 was a home computer produced by Sinclair Research,designed to be a low-cost introduction to home computing for the general public.

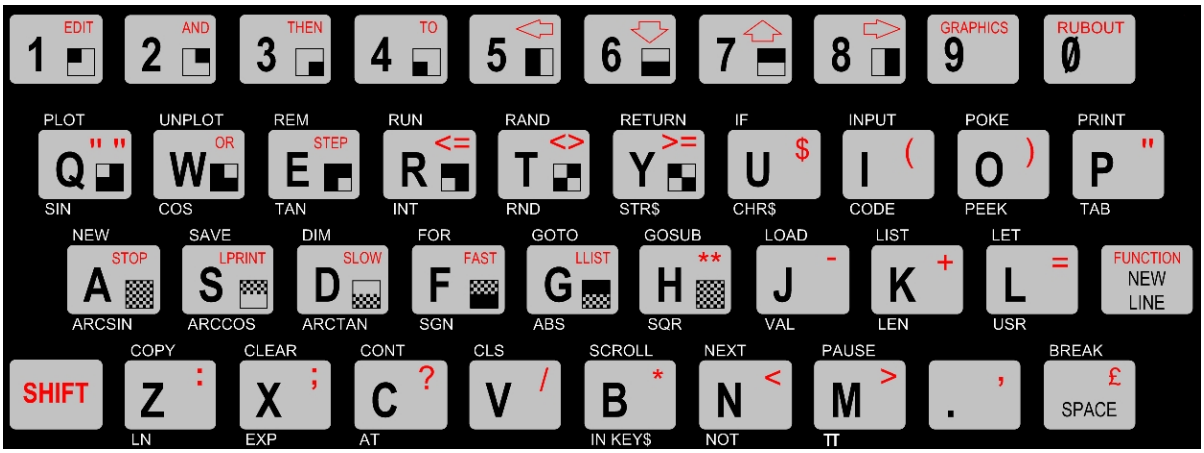
The ZXUNO+ version has been made by Jepalza based on a previous ZX97 implementation.

## SD card format

This core does not use the SD card

## Keyboard

The keyboard isn't mapped and the original machine keys layout is kept. For example, to obtain a " you have to type `Shift+P` or `Shift+0` to delete.




## Basic Guide

You can load a external tape (or from other external audio device) with the command `LOAD""`. Take note, that, while loading, the video output is disables and, unlike the original machine, you can hear the loading sound.

Some monitor stop playing audio if the video signal is lost. It's recommended to plug headphones or a external speaker if you want to hear the sound while loading a tape.

# ZX81 and ZX80

Alternative ZX81 and ZX80 core, made by avlixa, based on Grant Searle's ZX80 page.

Features:

- Selectable ZX80/ZX81 (ZX80 currently working only in RGB mode)
- 16k/32k/48k RAM packs
- 8KB with CHR$128/UDG addon (not tested)
- QS CHRS (not tested)
- CHROMA81
- Turbo in Slow mode: NoWait, x2, x8
- YM2149 sound chip (ZON X-81 compatible)
- Joystick types: Cursor, Sinclar, ZX81, ZXpand
- PAL/NTSC timings
- Turbo loading of .o and .p files
- Load alternative ROM
- Program loading using the audio input

## SD card format

You can use a SD card with the first partition in FAT16 or FAT32 format to store ROM and tape files.

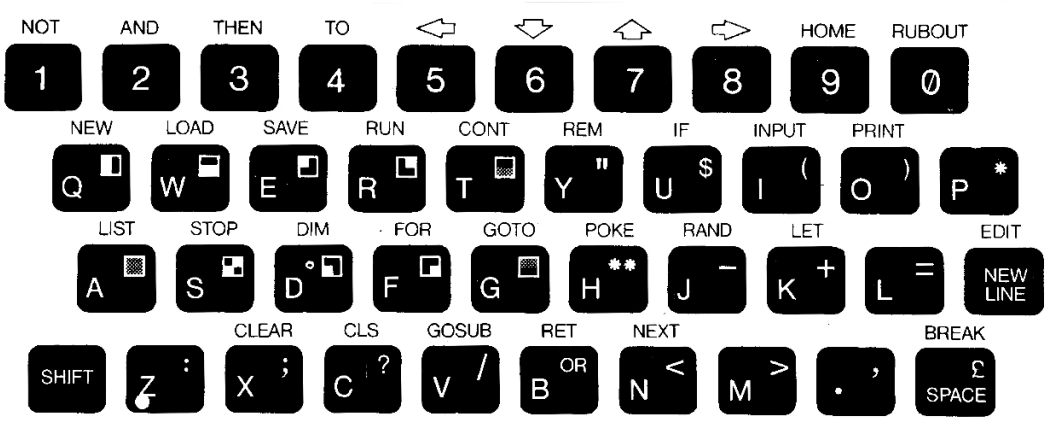
You can copy a file named `ZX8X.ROM` (available at the official repository into folder `/zx81/roms`: it is a concatenation of ZX81 rom (8k) + ZX80 rom (4k).

See the corresponding section for instructions of how to install the ZX81 and ZX80 core in ZXUNO+.

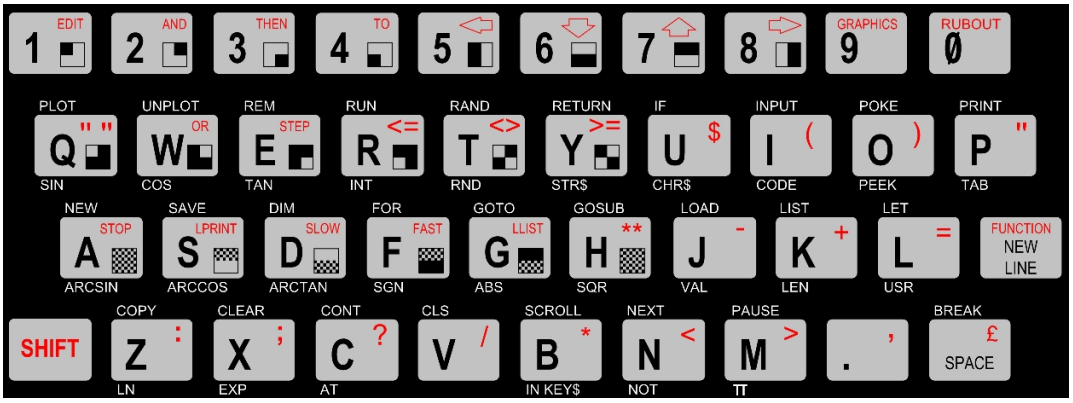
## Keyboard

The PS/2 keyboard isn't mapped and the original machine keys layout is kept. For example, to obtain a " you have to type `Shift+P` or `Shift+0` to delete.

### ZX80




### ZX81



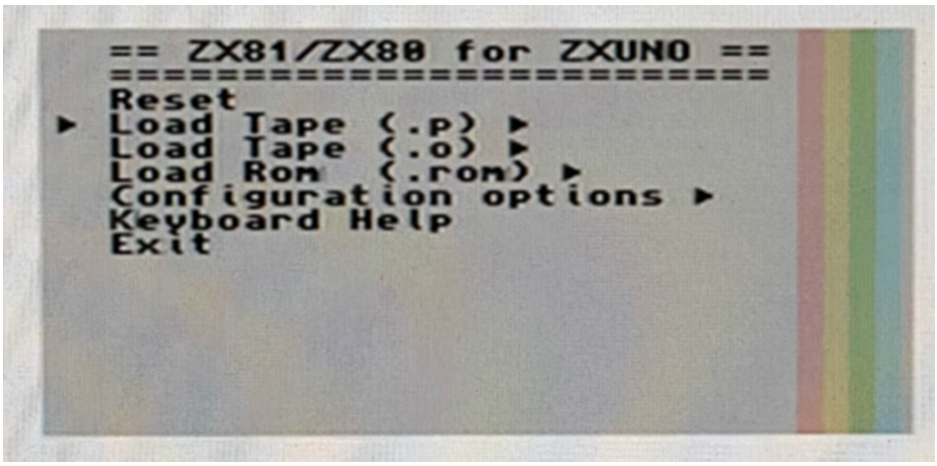

### Special keys and buttons

During the core execution:

- `F1` (`Caps Shift+Symbol Shift+1` on +UNO): Enable or disable the alternative chars
- `F5` (`Caps Shift+Symbol Shift+5` on +UNO) or joystick button 2: Show or hide configuration menu
- `F9` (`Caps Shift+Symbol Shift+9` on +UNO): Disables or enables the MIC audio output, since some games make annoying sounds when enabled
- `F10` (`Caps Shift+Symbol Shift+0` on +UNO): Enables or disables playing the audio input through the audio output, to hear loading sounds while loading
- `Scroll Lock` (`Caps Shift+Symbol Shift+G` on +UNO): Switch between RGB and VGA video output
- `Ctrl+Alt+Supr` (`Caps Shift+Symbol Shift+B` on +UNO): Reset
- `Ctrl+Alt+Backspace` (`Caps Shift+Symbol Shift+B` on +UNO): Hard reset

## Basic Guide

Pressing F5 (Caps Shift+Symbol Shift+5 on +UNO) or joystick button 2 shows or hides the configuration menu. Cursor keys (Caps Shift+5, Caps Shift+6, Caps Shift+7 and Caps Shift+8 on +UNO, PC XT keyboard mode) and Enter to select and choose menu options.




The following options are available:

- Reset
- Load Tape
- Load ROM
- Configuration Options
- Exit


- Computer Model: ZX80/ZX81
- Main RAM: 16K/32K
- Low RAM: Off/8KB
- Joystick: Cursor/Sinclair/ZX81
- QS CHRS: Disabled/Enabled
- CHR$128/UDG: 128 chars/64 chars/Disabled
- Chroma81: Disabled/Enabled
- Inverse Video: Off/On
- Black Border: Off/On
- Slow mode speed: Original/No Wait/x2
- Video frequency: 50Hz/60Hz

You can load a tape file selecting it from the menu, then enter the command `LOAD""` and `Enter`

Some monitors stop playing audio if the video signal is lost. It's recommended to plug headphones or a external speaker if you want to hear the sound while loading a tape. On a +UNO, the internal speaker will play the sound if nothing is connected to the audio out port.

`.p` files with colorization and char are supported.

For colorization to work, CHROMA81 should be enabled before loading. For alternate chars, QS CHRS should be enabled before loading.

The recommended options for most games are:

Main RAM: 16KB Low RAM: 8KB CHR$128: 128 chars QS CHRS: enabled CHROMA81: enabled

# Other Hardware

## ESPJoy

ESPjoy is an adapter board with DE-9 ports for Atari standard or Sega Genesis (Mega Drive) controllers that converts their button presses to equivalent keyboard press signals.

It is originally designed to use with ESPectrum, but can be used with other devices, as it is able to convert to two different keyboard protocols (USB and PS/2), with a built-in switch for selection.

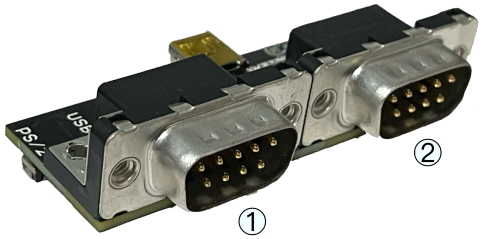
This keyboard translation is very useful for emulators or for FPGA-based devices, such as the ZX-Uno family and its derivatives or MiSTer, that do not have support for joystick connectors. For example, for computer cores when not using a permanently connected keyboard.

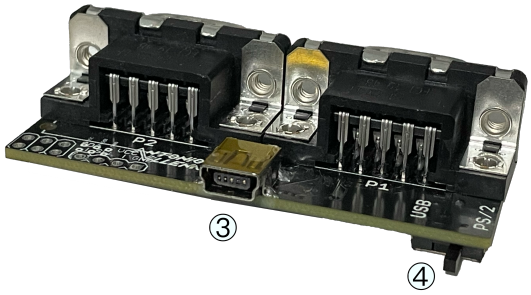
The on-board firmware sends keystrokes when a button of any connected controller is pressed and can use two controllers simultaneously.

One or two-button (Atari standard) controllers and joysticks are supported, as well as three or 6-button Sega controllers.

## Buttons and ports

| | |
|---|---|
| 1 | First DE-9 Port |
| 2 | Second DE-9 Port |




| | |
|---|---|
| 3 | USB Mini-B Port |
| 4 | Protocol selection switch |

## Use

### Configuration

To use with ESPectrum, the PS/2 option on the switch hast to be selected. The USB option can be used with MiSTer or another compatible system (PC, Mac, etc.). With the original firmware designed by David Carrión a HID keyboard is emulated, so it can be used without problem in MiSTer with computer cores.

Connection detection is performed in the first seconds after boot when the board is connected to a USB or PS/2 port is receiving power. If the board is connected to a device such as a PC or similar while starting up, it is very likely that the USB detection will fail, because the operating system is not yet ready when the board detection timeout is reached. In that case, it has to be disconnected and reconnected once the PC device is ready to detect the connection.

The PS/2 or USB mode selection switch hast to be set before connecting the board to power, or it will not function properly.

### Keymap selection

On startup, by pressing and holding any of the specified buttons, according to the tables below, you can select one of the different keyboard translations.

The default assignment behavior (when no button is pressed at startup) varies depending on the firmware you have installed on the ESPJoy. When the option (Original) is specified, we are talking about the firmware not adapted for ESPectrum. In the other case (ESPectrum), it's about ESPectrum adapted firmware.

The default assignment on ESPectrum firmware, when using the PS/2, port makes it incompatible with standard PS/2 devices. Therefore, in that case, you have to press a button which activates a different keyboard configuration during startup.

There are some clone controllers that can send strange signals in some directions. Because of this, the startup check in the code may fail, and the activation at startup, e.g. when pressing left on the keyboard, could not work properly.

## Port 1 (PS/2)

| Press on Start | Up | Down | Left | Right | A | B | C | Start | X | Y | Z | Mode |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| None (Original) | Up | Down | Left | Right | Escape | Enter | Right Alt | F1 | X | Y | Z | M |
| None (ESPectrum) | Up | Down | Left | Right | A | B | C | Start | X | Y | Z | Mode |
| Up | Q | A | O | P | Escape | M | Enter | F1 | X | Y | Z | C |
| Down | Up | Down | Left | Right | Escape | Enter | 0 | F5 | X | Y | Z | M |
| Left | 7 | 6 | 5 | 8 | Escape | 0 | Enter | F1 | X | Y | Z | M |

## Port 2 (PS/2)

| Press on Start | Up | Down | Left | Right | A | B | C | Start | X | Y | Z | Mode |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| None (Original) | Q | A | O | P | Escape | M | Enter | F1 | X | Y | Z | C |
| None (ESPectrum) | Up | Down | Left | Right | A | B | C | Start | X | Y | Z | Mode |
| Up | Up | Down | Left | Right | Escape | Enter | Right Alt | F1 | X | Y | Z | M |
| Down | Up | Down | Left | Right | Escape | Enter | 0 | F5 | X | Y | Z | M |
| Left | 7 | 6 | 5 | 8 | Escape | 0 | Enter | F1 | X | Y | Z | M |

## Port 1 (USB)

| Press on Start | Up | Down | Left | Right | A | B | C | Start | X | Y | Z | Mode |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| None (Original) | Up | Down | Left | Right | Escape | Enter | Right Alt | F12 | X | Y | Z | M |
| Up | Q | A | O | P | Escape | M | Enter | F12 | X | Y | Z | C |
| Down | Up | Down | Left | Right | Escape | Enter | 0 | F5 | X | Y | Z | M |
| Left | 7 | 6 | 5 | 8 | Escape | 0 | Enter | F12 | X | Y | Z | M |

## Port 2 (USB)

| Press on Start | Up | Down | Left | Right | A | B | C | Start | X | Y | Z | Mode |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| None (Original) | Q | A | O | P | Escape | M | Enter | F12 | X | Y | Z | C |
| Up | Up | Down | Left | Right | Escape | Enter | Right Alt | F12 | X | Y | Z | M |
| Down | Up | Down | Left | Right | Escape | Enter | 0 | F5 | X | Y | Z | M |
| Left | 7 | 6 | 5 | 8 | Escape | 0 | Enter | F12 | X | Y | Z | M |

## Firmware upgrade

There are different possible firmware versions (the program stored in the flash memory of the board and which does keystroke translating) that you can use:

- The original firmware, which can be used to send PS/2 or USB keystrokes, and is available here:

http://github.com/Dacarsoft/DB9_2_Keyboard

- The specific variant for ESPectrum, and which sends, by default, for PS/2, special emulator keystrokes, available here:

https://github.com/dacarsoft/DB9_2_Keyboard/tree/DB9_2_Keyboard_ESPectrum

- You can also install the Daemonbite firmware available at this location:

https://github.com/MickGyver/DaemonBite-Retro-Controllers-USB/tree/master/SegaTwoControllersUSB

In the latter case, the ESPJoy will function as a USB HID game controller, but without PS/2 protocol support.

To update the board program there are several software options, but all require a computer or similar device with USB ports and a Windows, Linux or macOS operating system.

The way to do the programming will depend on the format in which the program is available.

### Firmware as source code (INO format)

In this case, the firmware is not completely ready to be installed on the board and has to be compiled from its source code before being written to the board's flash memory. The best option in this case is to use the official Arduino development environment.

#### Arduino IDE

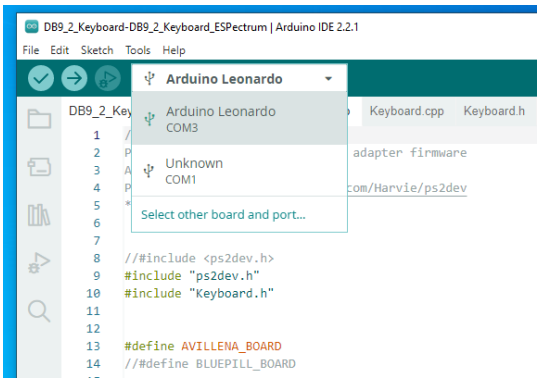
In order to do this type of installation, you can use a computer (Windows, Mac, Linux) with https://www.arduino.cc/en/software [Arduino IDE] installed.

### Update

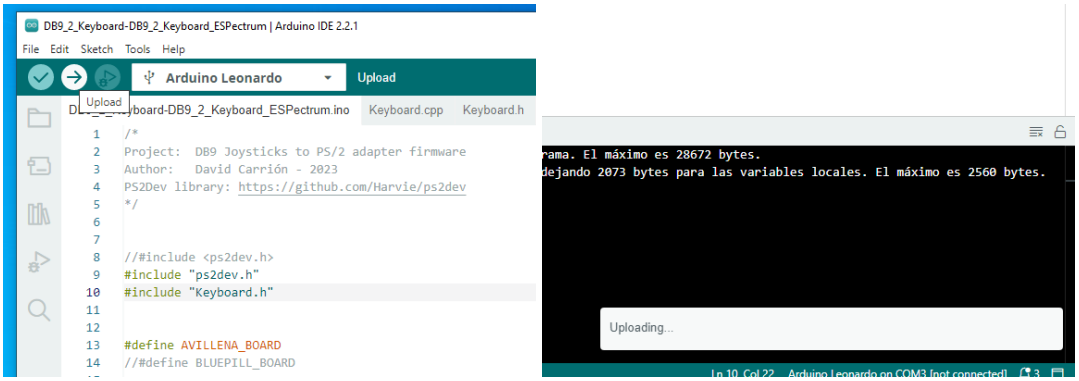
Once the environment is ready with Arduino IDE installed, download the desired version of the project from the corresponding repository (standard, ESPectrum, etc.). You should have a folder with a bunch of files, one of the with the `.ino` extension in its name.

Open the project file in Arduino IDE (for example `DB9_2_Keyboard_ESPectrum.ino`).

Set the switch on the board to the position marked with "USB" (the closest position to the USB Mini-B connector). Connect the ESPjoy to the computer using the USB cable and select it at the top of the IDE window ("Arduino Leonardo" option).



Finally, press the firmware upload button and wait a few seconds while the project is compiled and uploaded to the device.



### Firmware in HEX format

In this other case, the firmware is completely ready to be written to the board flash memory. There are multiple programs that can be used to do the programming.

#### AVRDUDE

AVR Downloader Uploader is a command line utility to download/upload/manipulate the ROM and EEPROM contents of multiple microcontrollers.

In order to use it with ESPJoy, you need a recent version that supports the `-r` parameter (reconnect to -P port after "touching" it). See the technical_notes at the end of this manual for more information.

The latest binary version for Windows can be downloaded from the official repository:

https://github.com/avrdudes/avrdude

For other systems, it is easy to compile the executable from the source code available in that repository. It is also available from multiple package installation systems, such as https://brew.sh/ [Homebrew for Mac], or the various official repositories of Linux distributions.

### Upgrade

After getting the desired `.hex` file firmware version from the (e.g. `DB9_2_Keyboard.ino.hex`), set the switch on the board to the position marked with "USB" (the closest position to the USB Mini-B connector). Then connect the ESPJoy board to a USB port of the computer , and identify that port in the operating system.

Once the port name has been identified, the file can be written to the board with a command such as the following:

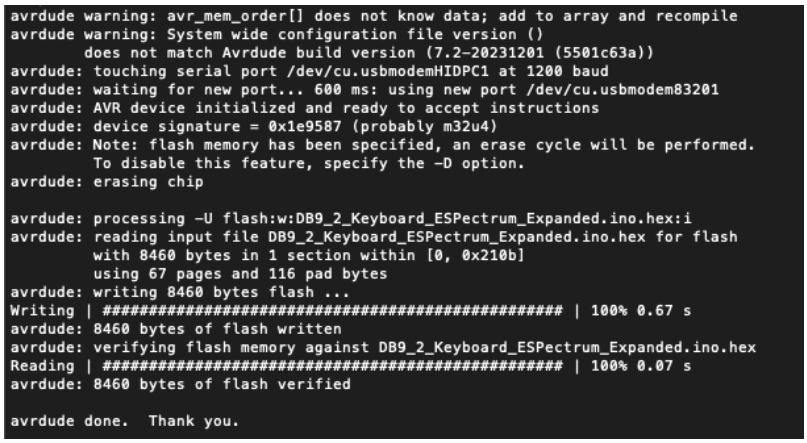
```
.../avrdude -F -patmega32u4 -cavr109 -b57600 -r -P <port> -U flash:w:<file.hex>:i
```

Where `<port>` is the name of the port where the board is connected and `<file.hex>` is the path to the `.hex` file you want to save.

For example, for a port named `/dev/cu.usbmodemHIDPC1` and a `DB9_2_Keyboard.ino.hex` file:

```
.../avrdude -F -patmega32u4 -cavr109 -b57600 -r -P /dev/cu.usbmodemHIDPC1 -U
flash:w:DB9_2_Keyboard.ino.hex:i
```

Wait for a few seconds, and, if all goes well, the console will display messages indicating that the program has been successfully recorded.



## Technical Notes

### About Arduino Leonardo programming

The ESPJoy board is based on a ATmega32u4 programmable controller, which is the one used by Arduino Leonardo boards.

These devices have a bootloader that can be used for flash programming, but can also be omitted to directly run the saved program. This is how you which one of the two possibilities is triggered:

- If it is started after a reset command (i.e. the board has a reset button that has been pressed), the programming mode (program saving in flash memory) is always activated.
- If there is no program stored in flash (the first word is `0xffff`), the programming mode is always activated.
- If it boots after a cold start (meaning that the device is activated for the first time after power is applied), it will try to execute the stored program.
- If it boots after a reset by watchdog, then:
  - If there is the appropriate code written to a magic location in SRAM (`0x7777` `written to` `0x0800`), then it will go into programming mode.
  - If not, it will try to execute the user code.
- If it boots after any other type of reset, it will go to programming mode.

The boot loader is designed to detect if a 1200 baud serial connection to the port has been opened and closed, and, if so, performs the necessary steps to perform a watchdog reset.

Finally, note that the bootloader has a timeout of 8 seconds. If programming has not started within that time after triggering (via any method) in programming mode, it will try again to execute the user code.

As a consequence of this, programming via commands, etc. with Windows requires specific drivers for both modes (program execution and programming) to be installed for all serial (COM) ports, so that the new connection after the restart by watchdog will work correctly.

When programming with AVRDUDE, it is highly recommended to use a version that has the `-r` parameter, and therefore automatically performs this port opening and closing process, and subsequent connection, within the 8 seconds margin required for programming. Otherwise it can be somewhat complex to do it manually.

# Rotary Encoders

The Pong core supports the use of quadrature rotary encoders as control devices. They can be connected to joystick ports. Although the testing has been done with 600 ppr encoders, lower ppr encoders, like 400 or 300, should also work.

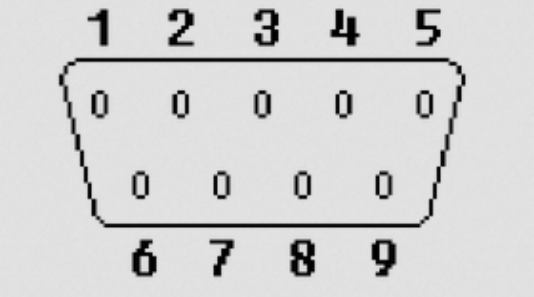
You can also use an Atari 2600 paddle driving controller. In this case the playing experience is bad, since they have few ppr and you must do several full rotations. When using them, it's recommended to set the accuracy setting to 8, to have enough speed.

## Connection

The joystick adapter must have pin 5 connected to positive VCC, used as main power, and pin 8 as GND. The rotary encoders to use must support voltage from 3,4v to 5v.

A rotary encoder has 5 wires: Earth Ground (not connected), Vcc (+), GND (0V`or `-), A and B.

A and B are connected to pins 1 and 2 for the first encoder, 3 and 4 for the second one. This way you can have up to 4 encoders connected using both joystick ports.




This way, the connections should be:

1. Line A encoder 1
2. Line B encoder 1
3. Line A encoder 2
4. Line B encoder 2
5. Vcc(+)
6. Fire 1
7. NC
8. GND
9. Fire 2

## Pong Core Configuration

Follow these directions to choose the configuration:

- For 1 or 2 endoders on joystick port 2 de joystick, select `1/2 Paddle in J2` option
- For 2 encoders, one for each joystick port, select `2/4 Paddle in J1&J2` option. This is also valid to connect two Atari 2600 driving paddles
- For 4 encoders, two for each joystick port, select `2/4 Paddle in J1&J2` option
- For 1 or 2 encoders on joystick port 2 along with a mouse (in this case the encoders are for players 2 and 4), select `Mouse PS/2` option

It is recommended to wait, and make the connection after selecting the chosen option, since the encoders interfere with the up/down directions of the joystick, blocking access to the menu. Another option is to add a on/off switch for the encoder that will disable the power.

# Loading from tape

Some cores can load, as the original machines could, from a external audio device like a cassette player or something else simulating it.

Besides the card, you have to plug an appropriate audio cable to ZXUNO+ audio input. It must have a 3.5 mm stereo jack on one side, and two mono outputs on the other side (one for each audio channel). The right audio mono is connected to the audio player (this is not necessary with a miniduino, since it already uses only the right audio channel when playing).

## Cassette Player

The use is exactly the same as when using the original computers:

1. Plug the audio cable
2. Type on the computer or select the tape loading option. For example, for ZX Spectrum 48K, typing `J`, then, twice, `"` and then `Enter` to do the classic `LOAD "" + Enter`
3. Start playing the tape (you may have to try several times adjusting the player volume)

## Computer

Depending on the operating system (Windows, macOS, Linux) there are several programs that can either play a tape file (`TAP`, `TZX`, `PZX`, etc.) and output the sound through a headphone output, or create an audio file (`WAV`, `VOC`, `AU`, etc.) that can be played using a music or audio program.

### PlayTZX

This program for Windows, macOS or Linux, can play directly a `TZX` tape file through the audio output of the computer.

You can download the binary file (for example), for Windows from World of Spectrum Classic and for Mac from this GitHub repository) or compile the source code as explained later.

1. Plug the audio cable between the computer audio output and ZXUNO+ audio input (remember to use only the right mono channel to the PC, Mac, etc. output)
2. Type on the computer or select the tape loading option. For example, for ZX Spectrum 48K, typing `J`, then, twice, `"` and then `Enter` to do the classic `LOAD "" + Enter`
3. Start playing a tape file with this command (you may have to try several times adjusting the player volume)

```
./playtzx <tape file path>
```

If everything works fine, you will see at the shell the name of the different tape data blocks, while the sound is played and the ZXUNO+ core loads the program.

On Linux, the program uses as output the device `/dev/dsp`, this may require to load a module like `snd_pcm_oss` (on systems using ALSA).

#### Compile source code (macOS or Linux)

To compile, the first thing is checking that the developer tools are installed on the system, including a C compiler (`gcc`, `clang`, command line developer tools for Mac, etc.) and GNU Autotools.

Download the source code from this repository), extract the contents if needed and access from a terminal to the directory and type the commands:

```
aclocal && autoconf && autoheader && automake --add-missing
./configure
make
```

If all goes well, a new file named `playtzx` will be created, which you can copy anywhere and then use. You can delete the compilation directory.

## Mobile phone, tablet, MP3 player, etc.

There are a very few apps (or none) that can directly play a tape file on a mobile device so, in many cases, the only option is to convert it to an audio file before playing it.

https://zxtape.net is a website, mobile device compatible, which can play TZP and TZX files

PlayZX is an App for Android which can play tape files through the headphone output.

The latest devices with headphone output are normally designed to work with impedances of only a few Ohms. This may, sometimes, not be enough for the ZXUNO+ audio input.

In these cases, it's recommended (if possible) to disable headphone volume limitations and/or use a headphone amplifier that can give a higher impedance.

### Audio file conversion

These are some of the many programs that exist and which can export tape files to audio files.

Tapir is a GUI program for Windows (but which can also run with Wine on Linux or Mac) that can load TZX and TAP files and export to WAV audio.

tape2wav from Fuse Utilities is a command line utility that can export from TZX PZX and TAP to WAV.

pzx2wav in PZX Tools is another command line utility which exports to WAV.

tsx2wav in TSXphpclass is made with PHP and that can export from TSX to WAV.

Audiotap is a graphical interface program made for Windows (but also working on Wine) that can convert Commodore TAP files to WAV audio files.

Comodoro 2.0 it's a script which can convert to TAP and TZX files for Commodore Pet, VIC-20, 16 and 64.

Lynx2Wav is a program that can convert Camputers Lynx TAP files to WAV audio.

2lynx2wav also can convert Camputers Lynx TAP files to audio files.

cgc2wav can convert EACA Colour Genie CAS files to WAV audio.

castool, available with MAME can convert many kinds of tape file data to WAV audio.

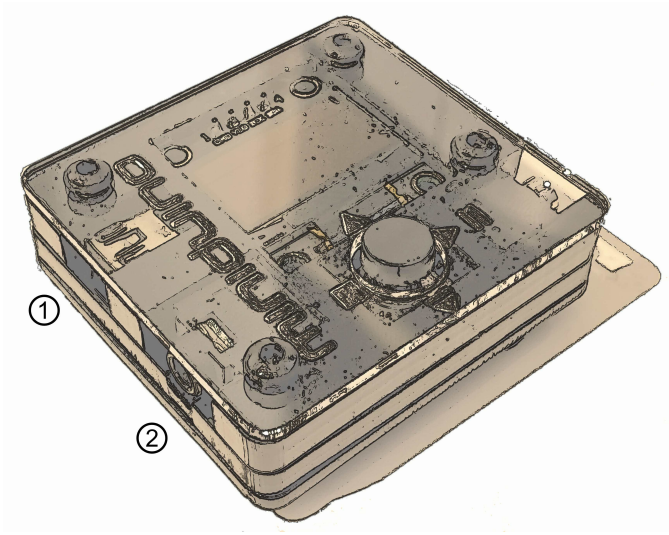
mzf2wav converts MZF tape files to WAV.

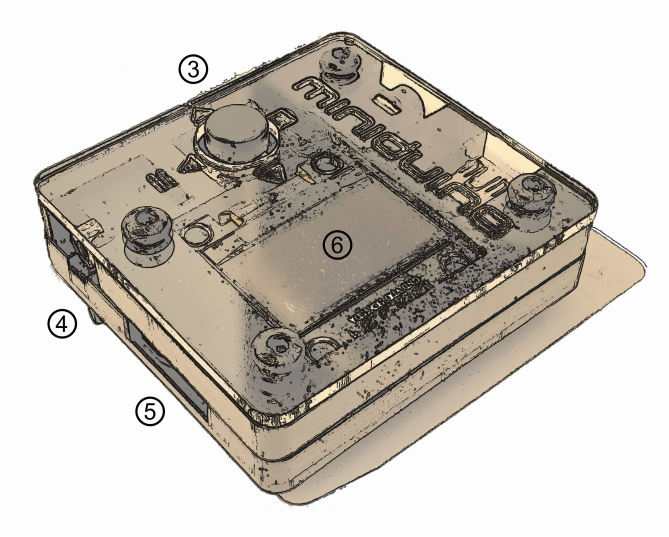
## Miniduino

Miniduino is a tape file audio player, based on an STM32F103C8T6 microcontroller or ATMega38P with the Maxduino firmware preinstalled.

Maxduino works in a similar way to cassette tape players, using digital tape files formats such as TAP and TZX (ZX Spectrum), O (ZX80), P (ZX81), CDT (Amstrad CPC), CAS(MSX) TSX (MSX, Acorn, etc). It's also possible to play AY music files as if they were tapes, to load them from SpecAY in a ZX Spectrum.

### Ports and buttons




| | |
|---|---|
| 1 | Power |
| 2 | Audio output |
| 3 | Control button |
| 4 | Motor control |
| 5 | SD card slot |
| 6 | Screen |

### Configuration

An SD card is required to store the tape files to play. Fast cards (Class 10 or greater) aren't recommended because there can be problems while reading the data. High capacity (SDXC or greater) cards aren't recommended either.

The SD card must have the first partition formatted as FAT16 or FAT32.

Besides the card, you must connect an appropriate audio cable to audio input. It must have a 3.5mm stereo jack on one side, and two mono output on the other side (one for each audio channel). The right audio mono is connected to the Miniduino.

If you have a device that can use motor control, you can also use a cable with a 2.6mm jack.

Copy the tape files (`TAP`, `TZX`, `O`, `P`, `CAS`, `TSX` and so on) to the first partition of the SD card. They can be organised using folders or directories.

The player shows file and folder entries in the order stored in the internal FAT table, not alphabetically. If you want to see them ordered you must reorder the SD card structure with a utility such as FATsort, YAFS, SDSorter or another application.

### Use

After the SD card with the data files is inserted, power it on by connecting the included USB power cable.


To show the optional menu, hold down the control button:

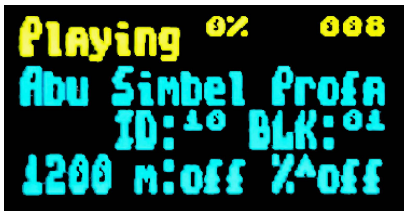
- Baud Rate: Configures turbo speed baud rates when playing 4B blocks in MSX files (`CAS` and `TSX`).
- Motor Ctrl: Enable this option when a control cable is connected to a proper device (Amstrad, CPC, MSX and so on).
- Converter (TSXCzxpUEFWS): Enables turbo loading `CAS` and `TSX` files, changes signal for Spectrum and Amstrad CPC files or change parity when playing Acorn Electron and BBC Micro `UEF` files.
- (Skip BLK)): To switch off (Skip ON) or enable an automatic pause when 2A blocks are found.

Outside the options menu, the control button is used as a four directional control joystick that has two different behaviours depending on whether the player is stopped or paused.


When the player is stopped (file and directories browser):

- Up and Down move through the current files and folders list.
- Left (Stop) goes one level up in the folder tree.
- Right (Play/Pause) enters a folder or, if the selection is a file, tries to play it.

When a file is playing you can stop it with the left button (Stop) or pause using the right button (Play/Pause).




When in pause (tape block browser:

- Up and Down move through the tape block files already played (useful for multiload titles or to load a previous level block).
- Left (Stop) cancels the player and goes back to file and folder browser mode.
- Right (Play/Pause) continues playing from the selected block.
- Press down the control button to enable or disable turbo mode for MSX.

### Making TZX or TSX files from other formats

While some tape file formats (Commodore, Camputers Lynx and so on) are not supported by Maxduino, there are some utilities that can, more or less successfully, embed audio data in a TSX or TZX file, which then can be used with Miniduino.

#### MakeTSX

You can use this command with NataliaPC's MakeTSX to create a TSX file with embedded audio:

```
...MakeTSX -b15 -wav audio_file.wav -tsx new_file.tsx
```

#### RetroConverter

Jorge Fuertes' RetroConverter can create a TZX file:

```
...retroconv audio_file.wav new_file.tzx
```

### Maxduino firmware upgrade

The Maxduino firmware is periodically updated and improved. You can track the changes and improvements either at the Va de Retro forums or at the GitHub project page. To take advantage of this improvements, the Miniduino flash image must be flashed with the updated firmware version.

There are two Miniduino models; one based on the STM32 microcontroller and the ther on the ATMega328P.

#### STM32 Model

### Environment setup

Firmware flashing is done from a computer (Windows, macOS or Linux) with Arduino IDE installed.

You must install SDFat (1.1.4) software library selecting the menu option Program → include library → manage libraries.

Minidiuno microcontroller support must also be added. This is done in two steps:

1. Add ARM Cortex M3 support from menu Tools → board → board manager, and installing "Arduino SAM boards (Cortex-M3)".
2. Add STM32 microcontroller support; download the file available at this link.

Extract the contents to the current user folder in:

```
...Arduino/hardware/Arduino_STM32
```

On Windows, install the USB device controller running (with elevated privileges:

```
 ...\drivers\win\install_drivers.bat
```

On Linux, install with root privileges the necessary `udev` rules:

```
...tools/linux/install.sh
```

On macOS, if Miniduino doesn't appear as a USB device in Arduino ID when connected it may be necessary to install libusb.

Last, on Mac or Linux, the file `maple_upload` inside `Arduino_STM32` must be edited with a text editor. Those lines do not work:

```
if [ $# -eq 5 ]; then
    dfuse_addr="--dfuse-address $5"
else
    dfuse_addr=""
fi
```

They must be changed to:

```
dfuse_addr=""
```

### Upgrade

After the environment is ready, download the software from the official repository in GitHub.

The Miniduino player with STM32 microcontroller is only supported from version 1.65 and up.

Load the project file with Arduino IDE (for example `MaxDuino_v1.69.ino`).

Ensure that all logo entries in `userSTM32Config.h` file are commented out except for Miniduino.

```
...
//#define tanque4
//#define tanque1
//#define dostanques
//#define cablemax
//#define sony
#define miniduino
...
```

Connect the Miniduino device to the computer using the USB cable, and find the assigned port, typically called something like "Maple Mini" (for example: COM5 Maple Mini)

Fom the "Tools" menu set these options:

```
Board: Generic STM32F103C Series.
Variant: STM32F103C8 (20k RAM, 64k Flash).
Upload Method: STM32duino bootloader.
CPU Speed: 72Mhz (Normal).
Optimize: Smallest (default).
Port: <Previously identified port>.
```

Last, click on the firmware load button and wait for a few seconds while the project is compiled

and loaded into the device.

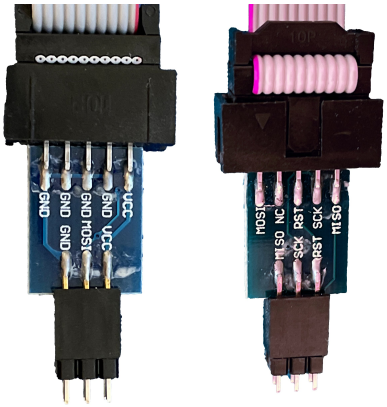
If everything has been done correctly the Miniduino will restart and show on the screen the new firmware version.

#### ATMega328P Model

### Environment setup

**Requirements**:

- One hex key with the right socket size for the cover screws
- USBasp flash programmer




Also, firmware flashing is done from a computer (Windows, Mac, Linux) with Arduino IDE installed.

You must install SDFat (1.1.4) software library selecting the menu option Program → include library → manage libraries.

### Upgrade

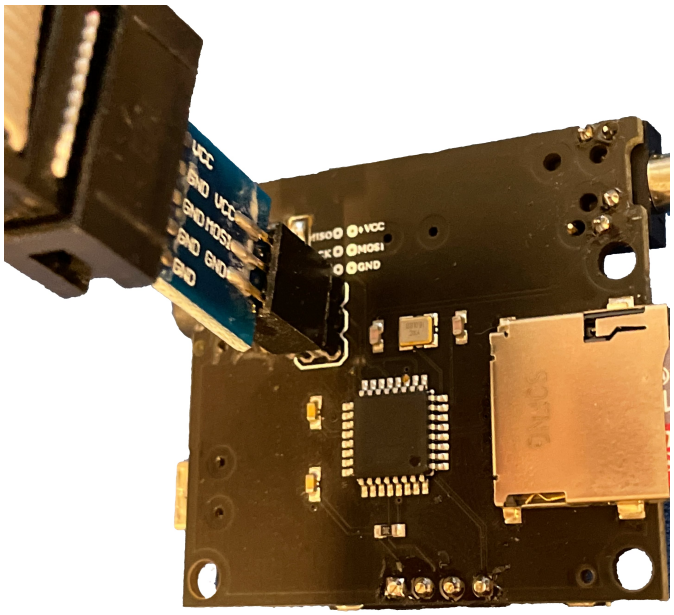
After you have the environment ready, download the software from the official repository in GitHub

Load the project file with Arduino IDE (for example `MaxDuino_v1.69.ino`).

Check in the file `userconfig.h` that all logo entries are commented except for Miniduino and, if not, change them.

```
...
//#define tanque4
//#define tanque1
//#define dostanques
//#define cablemax
//#define sony
#define miniduino
...
```

Connect the Miniduino device to the USBasp programmer, making sure that the connector is in the right position (i.e VCC with VCC, MOSI with MOSI, GND witch GND, etc.), and connect the USB adapter to the computer




Set the following options in the "Tools" menu:

```
Board: Arduino Pro or Pro Mini
Processor: ATmega328P (5V,16 Mhz)

Programmer: "USBasp"
```

Last, hold down the 'Shift' key on the computer while clicking the firmware load button. Wait for a few seconds until the project is compiled and loaded into the device.

If everything was done correctly the Miniduino will restart and show on the screen the new firmware version.

# Troubleshooting

## Firmware images management

There are several tools with you can use to make and/or edit the contents of `ZX1`, `ZX2`, `ZXD` files.

### zx123_tool

This is a tool to analyze, extract and inject data in SPI flash image files for ZX-Uno, ZXDOS and similar devices.

You need to have Python 3 to use it. Depending on the operating system you may have to install it.

Having Python 3, you only need to download the latest version of the tool from the official repository, following this link.

Once extracted, you have to run from a shell the main script using Python 3. This may change depending on the operating system.

For example, on Windows, it's usually:

```
py -3 zx123_tool.py
```

With other operating systems it normally is like:

```
python3 ./zx123_tool.py
```

You also need a SPI flash image file. This can be obtained from within the Spectrum core in "root" mode, with one of the commands `back16m`, `backzx2` or `backzxd`. Once you have obtained the extracted file from the SD, you can "clean" it leaving only the Spectrum core and the first Spectrum ROM with a command like this:

```
... zx123_tool.py -i FLASH.ZX1 -w -o FLASHempty.ZX1
```

Where `FLASH.ZX1` is the path to the original file and `FLASHempty.ZX1` is the path to the new "clean" file.

### List the contents of an image

To see the contents of an image file named `FLASH.ZX1` (installed cores and some configuration info), you can use the command

```
... zx123_tool.py -i FLASH.ZX1 -l
```

To show the contents of the same file, including ZX Spectrum ROMs info:

```
... zx123_tool.py -i FLASH.ZX1 -l -r
```

### Change the BIOS of an image

To change the BIOS inside a file named `FLASH.ZX1`, using the BIOS file named `FIRMWARE.ZX1`

```
...zx123_tool.py -i FLASH.ZX1 -a BIOS,FIRMWARE.ZX1
```

You can, at the same time, modify some of the default parameters. For example, with the options; `-m` for video mode: 0 (PAL), 1 (NTSC) or 2 (VGA), `-k` for the keyboard layout: 0 (Auto), 1 (ES), 2 (EN) or 3 (Spectrum).

This way to change the BIOS of a file named `FLASH.ZX1`, using the BIOS file `FIRMWARE.ZX1`, and also set the video mode to VGA:

```
...zx123_tool.py -i FLASH.ZX1 -a BIOS,FIRMWARE.ZX1 -m 2
```

There are also options to set the BIOS boot delay time, the default core or the default Spectrum ROM. See the documentation for more info.

### Add a Spectrum ROM to an image

To add a Spectrum ROM file named `48.rom`, with the name `Spec48` and using the slot number 5, you can use a command like:

```
...zx123_tool.py -i FLASH.ZX1 -a ROM,5,xdnlh17,Spec48,48.rom
```

See the documentation for all the possible options when adding a Spectrum ROM.

Amongst the information you give when adding a ROM, there are some flags used to tell which hardware options, etc, you want to have enabled or disabled when loading the ROM, as shown in this table:

| i | **Keyboard issue 3 enabled (instead of issue 2)** |
|---|---|
| c | Disable memory contention |
| d | Enable DivMMC |
| n | Enable NMI DivMMC (esxdos Menu) |
| p | Use Pentagon Timings |
| t | Use 128K timings |
| s | Disable DivMMC and ZXMMC ports |
| m | Enable Timex Horizontal MMU |
| h | Disable ROM high bit (1FFD bit 2) |
| l | Disable ROM low bit (7FFD bit 4) |
| 1 | Disable 1FFD port (+2A/3 paging) |
| 7 | Disable 7FFD port (128K paging) |
| 2 | Disable TurboSound (secondary AY chip) |
| a | Disable AY chip |
| r | Disable Radastanian mode |
| x | Disable Timex mode |
| u | Disable ULAPlus |

### Install a Core to an image

For example, to install a core in space 3, from a file named TBBLUE.ZX1, with the name TBBlue, use a command like this:

```
...zx123_tool.py -i FLASH.ZX1 -a 'CORE,3,TBBlue,TBBLUE.ZX1'
```

If you want also to set the core as the default, use a command like:

```
...zx123_tool.py -i FLASH.ZX1 -a 'CORE,3,TBBlue,TBBLUE.ZX1' -c 3
```

### Change esxdos ROM of an image

Just like the BIOS firmware, you can install a ROM esxdos file, with a command like this:

```
...zx123_tool.py -i FLASH.ZX1 -a esxdos,ESXMMC.BIN
```

### Mix several actions in one line

Please do note that you can add several actions in one command line. For example, to "clean" an image file named FLASH.ZX1, creating a new one named FLASHnew.ZX1, installing the BIOS from the file FIRMWARE.ZX1, set up video mode to VGA, the keyboard in Spectrum mode, add a Spectrum ROM file 48.rom, with the name Spec48 while using slot number 5, install a core at space 3, from a file named TBBLUE.ZX1, with the name TBBlue, as default core:

```
... zx123_tool.py -i FLASH.ZX1 -w -o FLASHnew.ZX1 -a BIOS,FIRMWARE.ZX1 -m 2 -k 3 -a
ROM,5,xdnlh17,Spec48,48.rom -a CORE,3,TBBlue,TBBLUE.ZX1 -c 3
```

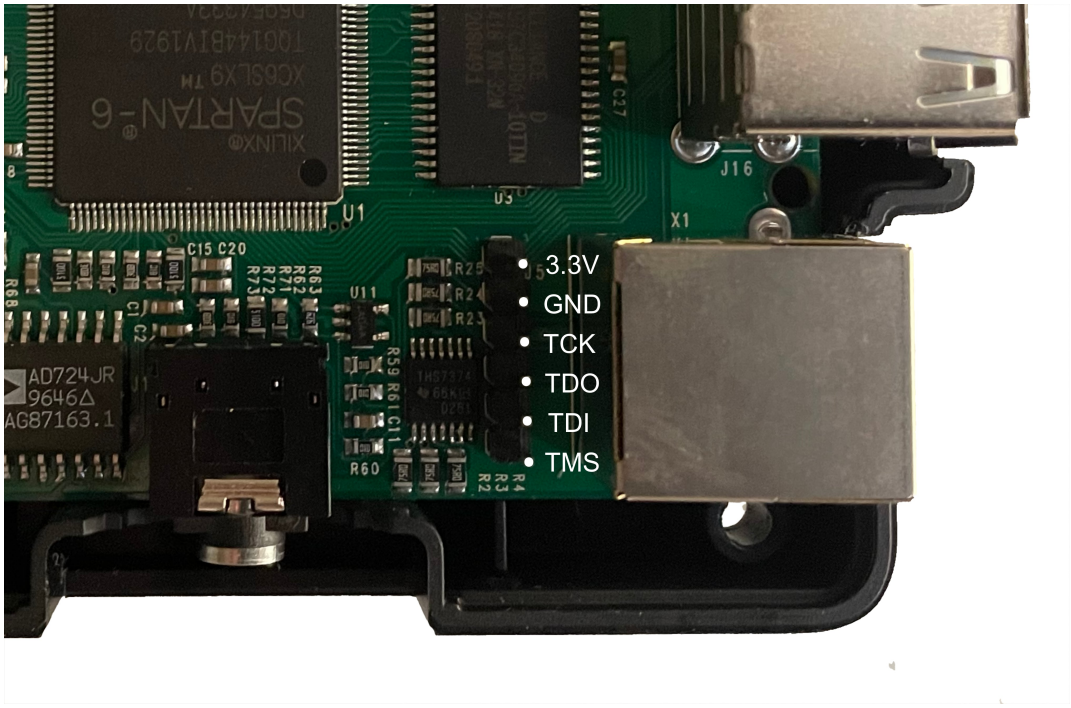
# Firmware recovery

Sometimes (e.g. when installing an experimental core or when upgrading the ZX Spectrum Core or the BIOS) it may happen that the ZXUNO+ stops booting. The board LEDs are on, but there is no display, and it doesn't do anything when trying the different key combinations to access BIOS setup, etc.

When this happens, there are several recovery methods that let you install again the firmware.

## JTAG cable connections

Later, in some of the recovery steps, when talking about jump wires or USB-Blaster connections to the ZXUNO+ board, you can use these images as reference.




**DO NOT** connect the 3V line

For a +UNO, you have to unplug the ZX-Uno board to make a recovery.

## Recovery using a Raspberry Pi

**Hardware required**:

- Raspberry Pi (with SD card, keyboard, display, power supply, etc.) and with internet connection
- 5 jump wires (if possible, female on both sides) or, instead a USB-Blaster cable
- One screwdriver for ZXUNO+ cover screws
- SD for ZXUNO+ with the first partition formatted as FAT16 or FAT32
- Keyboard and display for ZXUNO+

**Software required**:

- Flash image and recovery file for ZXUNO+ from Github repository and ZX-Uno forum

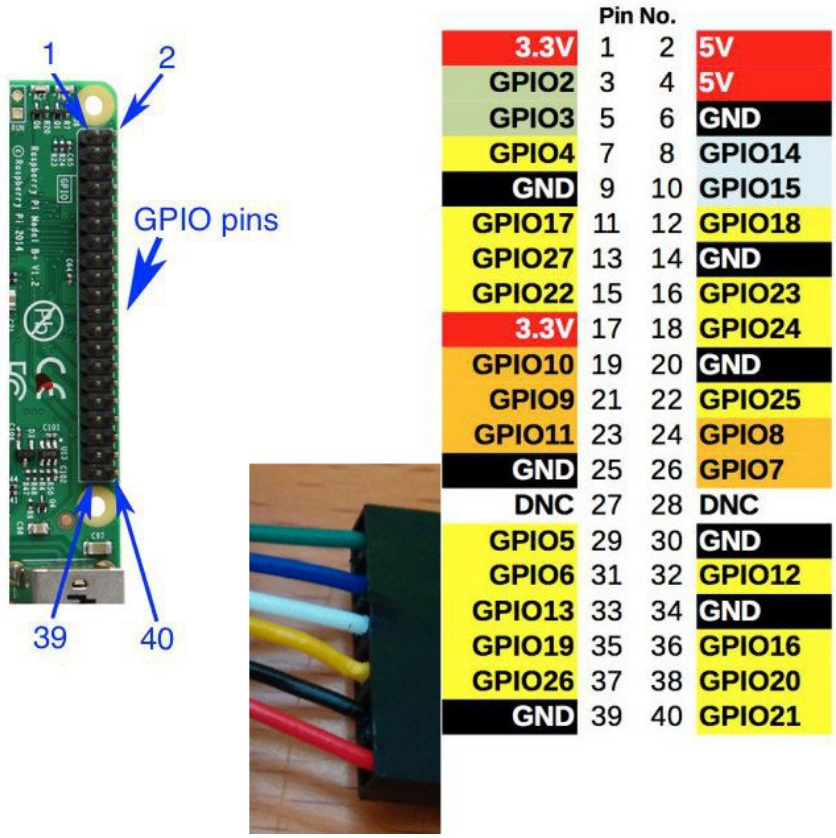
**Instruction Steps**:

1. Install Raspberry Pi OS (formerly known as Raspbian) to the Raspberry Pi SD card (using the official download, NOOBS, PINN, etc.)
2. Install Open OCD:

```
sudo apt-get update
sudo apt-get install git autoconf libtool make pkg-config
sudo apt-get install libusb-1.0-0 libusb-1.0-0-dev telnet
sudo apt-get install libusb-dev libftdi-dev
git clone git://git.code.sf.net/p/openocd/code openocd-code
cd openocd-code/
./bootstrap
./configure --enable-usb_blaster --enable-sysfsgpio --enable-bcm2835gpio
make
sudo make install
cd ..
rm -rf ./openocd-code
```

3. Connect USB-Blaster or jump wires if using GPIO. In this case, open the ZXUNO+ case and, as explained before connect the FPGA JTAG lines (TMS, TDI, TDO, TCK and GND), using the wires, to the Raspberry Pi GPIO pins.

Is using a GPIO connection, take note of the chosen pins, making sure that GND is connected with GND.




In this example, the 31, 33, 35, 37 and 39 pins will be used (corresponding to GPIO #6, GPIO #13, GPIO #19, GPIO #26 and GND), like this:

| ZXUNO+ JTAG | GPIO | Raspberry Pi Pin |
|---|---|---|
| TMS | GPIO#6 | 31 |
| TDI | GPIO#13 | 33 |
| TDO | GPIO#19 | 35 |
| TCK | GPIO#26 | 37 |
| GND | GND | 39 |

4. Copy to the Raspberry Pi the file named recovery.bit previously downloaded from the ZX-Uno forum. For our example, it will be at /home/pi/zxunoplus/unbrick/
5. If using GPIO, make a copy of Open OCD configuration file, to the same directory where recovery.bit is.

```
cp /usr/local/share/openocd/scripts/interface/raspberrypi2-native.cfg
/home/pi/zxunoplus/unbrick/
```

6. For GPIO connection, edit `raspberrypi2-native.cfg` copy, updating `bcm2835gpio_jtag_nums` (uncommenting, if necessary), with your JTAG and GPIO connection numbers, at the line `bcm2835gpio_jtag_nums`. For our example:

```
# Header pin numbers: 37 31 33 35
bcm2835gpio_jtag_nums 26 6 13 19
```

7. Comment, if it wasn't already, the line `bcm2835gpio_swd_nums` (not necessary for USB-Blaster connection):

```
#bcm2835gpio_swd_nums 11 25
```

8. Add, to the end of the file, the line `adapter speed 250` (again, not necessary for USB-Blaster):

```
adapter speed 250
```

9. Turn on the ZXUNO+.
10. Make sure that, on the Raspberry Pi, we are in the directory where `recovery.bit` is, and execute the command that loads the BIOS on recovery mode, using the path to the previously edited `raspberrypi2-native.cfg`.

For GPIO connection:

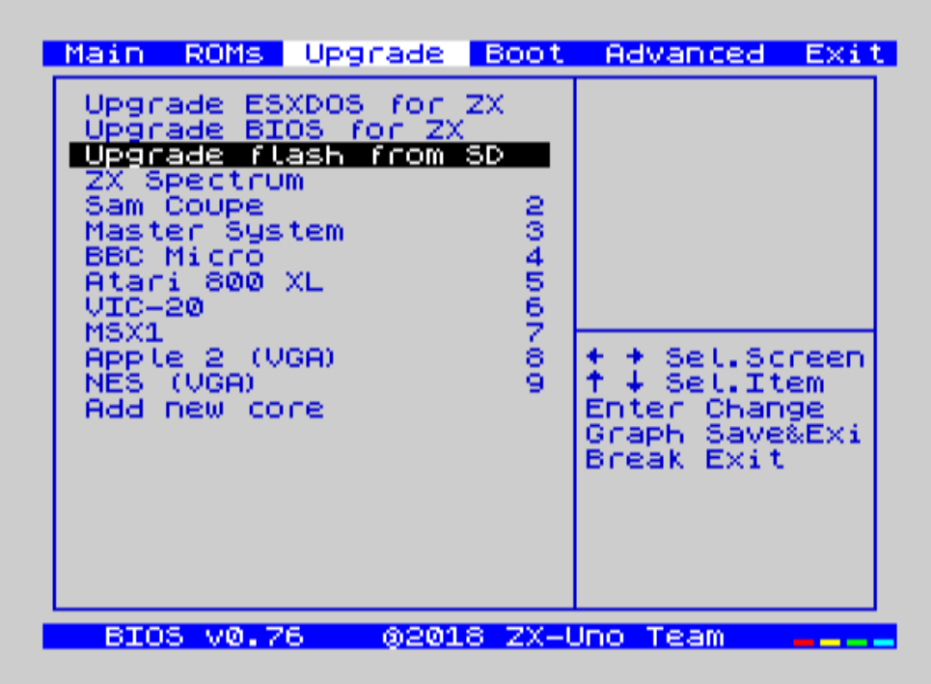
```
cd /home/pi/zxunoplus/unbrick
sudo openocd -f /home/pi/zxunoplus/unbrick/raspberrypi2-native.cfg -f
/usr/local/share/openocd/scripts/cpld/xilinx-xc6s.cfg -c "init; xc6s_program xc6s.tap;
pld load 0 recovery.bit ; exit"
```

For USB-Blaster connection:

```
sudo openocd -f /usr/local/share/openocd/scripts/interface/altera-usb-blaster.cfg -f
/usr/local/share/openocd/scripts/cpld/xilinx-xc6s.cfg -c "init; xc6s_program xc6s.tap;
pld load 0 recovery.bit ; exit"
```

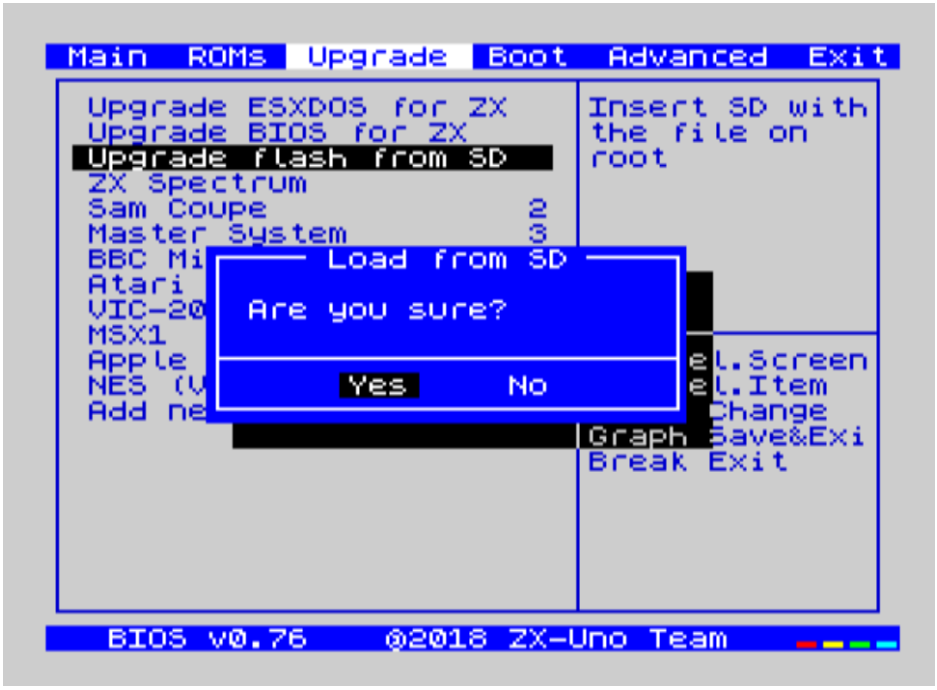
11. If all goes well, we will see that the FPGA LED change their state and the BIOS is shown on the display.

If there is no image on the display, press `Scroll Lock` (`Caps Shift+Symbol Shift+G` on +UNO) to switch between RGB and VGA modes, just in case the recovery BIOS did start in the wrong mode for our setup.




12. Insert in the ZXUNO+ the SD card formatted as FAT16 o FAT32, and with the `FLASH.ZX1` file downloaded previously.
13. If using a USB-Blaster connection, unplug the connector.

14. Select the option `Upgrade Flash from SD`. Press Enter, choose `Yes`, and press Enter again to start the Flash writing process.




This process can't be undone, and it will replace all the previously installed cores, the BIOS, the ZX Spectrum ROMs and their configuration with the data in the image file.

Usually, the recovery image is set to use a PS/2 keyboard so, for a +UNO, some key combinations, like `Caps Shift + 5`, etc may not work. In this case, you have to change the keyboard mode to `PC XT`(`Caps Shift + Symbol Shift + U` and then `9`), to make them work temporarily.

15. After some minutes, the process will end, and, after turning the ZXUNO+ off and on, it should start fine.

If no image is shown, press again `Scroll Lock` (`Caps Shift+Symbol Shift+G` on +UNO) to switch between RGB and VGA modes. In this case, you should have to enter the BIOS and change the right advanced setting that matches your display.

For a +UNO, since the recovery image uses a PS/2 configuration as default, follow this steps to set up the BIOS correctly:

1. If you see no image, switch between RGB and VGA mode (`Caps Shift+Symbol Shift+G`)
2. Change to `PC XT` keyboard mode (`Caps Shift+Symbol Shift+U` and then `9`)
3. Reboot +UNO without losing the temporary keyboard mode (`Caps Shift+Symbol Shift+B`)
4. Quickly, press `Caps Shift + 1`
5. Again, if there's no image, switch between RGB and VGA mode (`Caps`

Shift+Symbol Shift+G)

6. Navigate through BIOS and turn on these options:
   - Advanced → Keyboard Layout: Spectrum
   - Advanced → Video: VGA (only if there was no image)
7. Save changes:
   - Exit → Save changes and exit
8. Completely turn off the +UNO and turn it on again

If no image is shown, press again Scroll Lock (Caps Shift+Symbol Shift+G on +UNO) to switch between RGB and VGA modes. In this case, you should have to enter the BIOS and change the right advanced setting that matches your display.

## Recovery using macOS and USB-Blaster cable

**Hardware required**:

- USB-Blaster cable. Using the right pins for ZXUNO+, as explained earlier
- Flash Image file and and recovery file for ZX-Uno. The same ones as for Raspberry Pi, from Github repository and ZX-Uno forum

**Software required**:

- Mac OS
- Extra data folder for UrJTAG, obtained from here
- Homebrew for Mac OS
- UrJTAG: following instructions from here:

**Instruction Steps**:

1. Prepare UrJTAG instal:

```
brew install libftdi libusb pkg-config
git clone https://github.com/C-Elegans/urjtag.git
cd urjtag
```

2. Copy the extra data folder for UrJTAG, into urjtag `data` folder.
3. Start compiling:

```
./configure --with-libftdi --with-libusb --with-ftd2xx --with-inpout32 --enable-
python=no
make -j4
sudo make install
```

4. Copy `FLASH.ZX1` file to the root of the ZX-Uno SD card.
5. Connect USB-Blaster cable to ZXUNO+ and the Mac
6. Turn on the ZXUNO+ or +UNO.
7. Make sure that, we are in the directory where `recovery.bit` is, and execute `jtag` command.

8. A new shell appears. Now type the commands:

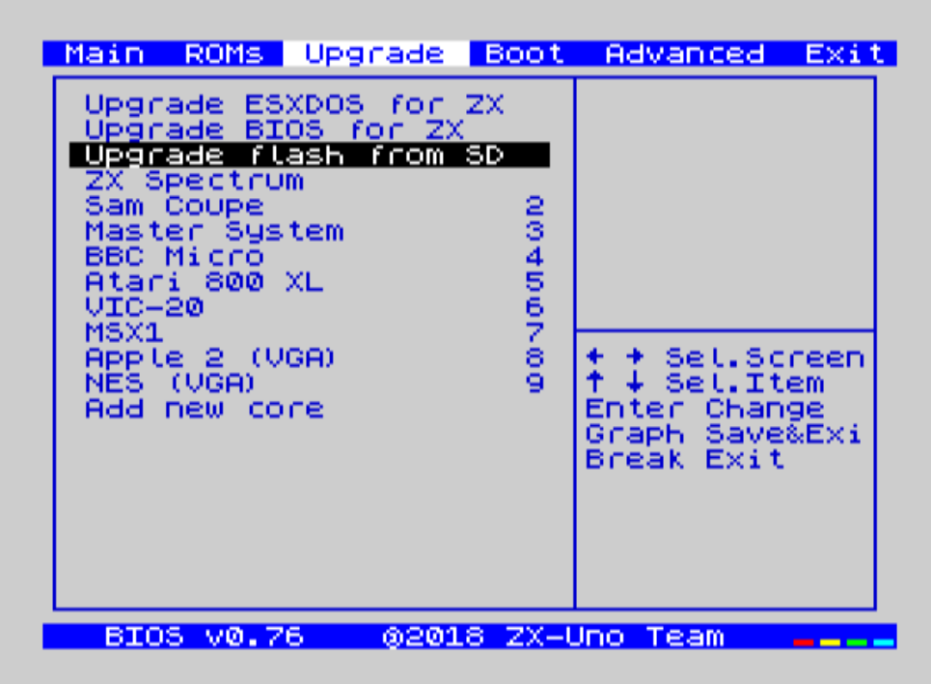
```
cable usbblaster
detect
pld load recovery.bit
```

Make sure, when using `detect`, that the device is shown as detected. You may need to execute `detect` several times until it appears.

```
UrJTAG 2018.06 #
Copyright (C) 2002, 2003 ETC s.r.o.
Copyright (C) 2007, 2008, 2009 Kolja Waschk and the respective authors

UrJTAG is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
There is absolutely no warranty for UrJTAG.

warning: UrJTAG may damage your hardware!
Type "quit" to exit, "help" for help.

jtag> cable usbblaster
Connected to libftdi driver.
jtag> detect
IR length: 6
Chain length: 1
Device Id: 00100100000000000100000010010011 (0x24004093)
  Manufacturer: Xilinx (0x093)
  Part(0):      xc6slx25 (0x4004)
  Stepping:     2
  Filename:     /usr/local/share/urjtag/xilinx/xc6slx25/xc6slx25
jtag> pld load recovery.zxd.bit
Bitstream information:
        Design: tld_zxuno_lx16.ncd;UserID=0xFFFFFFFF
        Part name: 6slx25ftg256
        Date: 2020/05/29
        Time: 01:32:07
        Bitstream length: 801462
jtag>
```

9. If all goes well, we will see that the FPGA LED change their state and the BIOS is shown on the display.

If there is no image on the display, press `Scroll Lock` (`Caps Shift+Symbol Shift+G` on +UNO) to switch between RGB and VGA modes, just in case the recovery BIOS did start in the wrong mode for our setup.




10. Insert in the ZXUNO+ the SD card formatted as FAT16 o FAT32, and with the `FLASH.ZX1` file downloaded previously.
11. Unplug the USB-Blaster.

12. Select the option `Upgrade Flash from SD`. Press Enter, choose `Yes`, and press Enter again to start the Flash writing process.


This process can't be undone, and it will replace all the previously installed cores, the BIOS, the ZX Spectrum ROMs and their configuration with the data in the image file.

Usually, the recovery image is set to use a PS/2 keyboard so, for a +UNO, some key combinations, like `Caps Shift + 5`, etc may not work. In this case, you have to change the keyboard mode to `PC XT`(`Caps Shift + Symbol Shift + U` and then `9`), to make them work temporarily.

13. After some minutes, the process will end, and, after turning the ZXUNO+ off and on, it should start fine.

If no image is shown, press again `Scroll Lock` (`Caps Shift+Symbol Shift+G` on +UNO) to switch between RGB and VGA modes. In this case, you should have to enter the BIOS and change the right advanced setting that matches your display.

For a +UNO, since the recovery image uses a PS/2 configuration as default, follow this steps to set up the BIOS correctly:

1. If you see no image, switch between RGB and VGA mode (`Caps Shift+Symbol Shift+G`)
2. Change to `PC XT` keyboard mode (`Caps Shift + Symbol Shift + U` and then `9`)
3. Reboot +UNO without losing the temporary keyboard mode (`Caps Shift + Symbol Shift + B`)
4. Quickly, press `Caps Shift + 1`
5. Again, if there's no image, switch between RGB and VGA mode (`Caps`

Shift+Symbol Shift+G)

6. Navigate through BIOS and turn on these options:
   - Advanced → Keyboard Layout: Spectrum
   - Advanced → Video: VGA (only if there was no image)
7. Save changes:
   - Exit → Save changes and exit
8. Completely turn off +UNO and turn it on again

When using Linux with urjtag, the process should be quite similar, although the dependencies (libftdi libusb pkg-config) would have to be installed with the appropriate package manager (apt, yum, pacman, etc.)

## Recovery using Windows and USB-Blaster cable

**Hardware required**:

- USB-Blaster cable. Using the right pins for ZXUNO+, as explained earlier
- Flash Image file and and recovery file fo ZXUNO+. The same ones as for Raspberry Pi, from GitHub repository and ZX-Uno forum

**Software required**:

- Windows
- USB-Blaster drivers for Windows, available at ZX-Uno forums
- UrJTAG, for Windows, available at the official repository

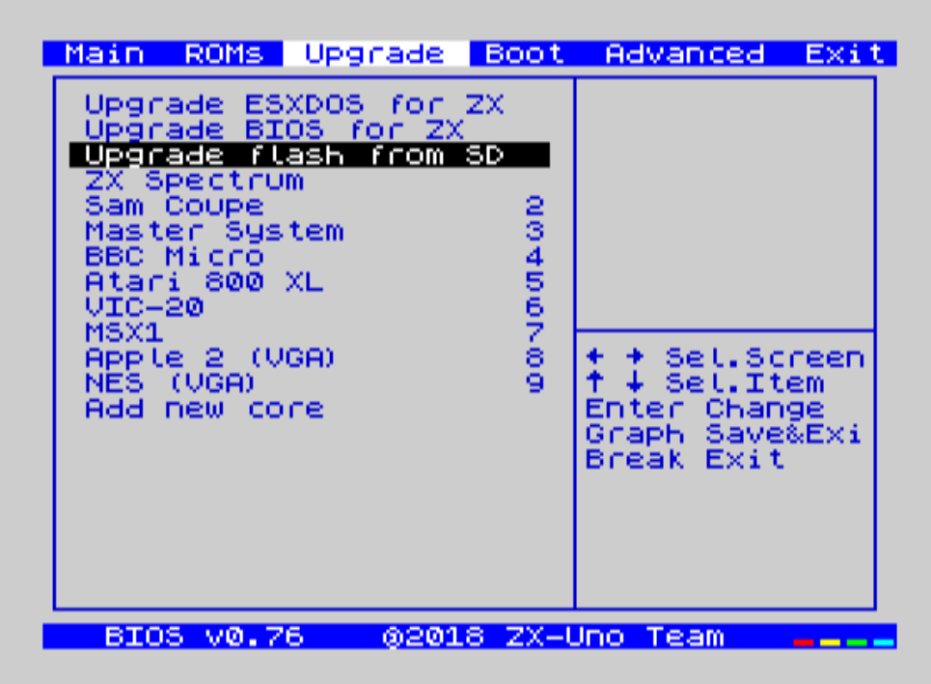
**Instruction Steps**:

1. Extract the ZIP file with drivers (obtained from the ZX-Uno forums)
2. Plug USB-Blaster to the Windows PC and install the driver choosing to select manually installation files and using the folder `drivers` obtained after extracting the ZIP
3. Extract UrJTAG Windows software, obtained at the official repository
4. Copy `FLASH.ZX1` file to the root of the ZX-Uno SD card.
5. Connect USB-Blaster cable to ZXUNO+ and the PC
6. Make sure that you are in the directory where `recovery.bit` is, and execute `jtag` command.
7. Turn on the ZXUNO+ or +UNO.
8. A new shell appears. Now type the commands:

```
cable usbblaster
detect
pld load recovery.bit
```

9. If all goes well, we will see that the FPGA LED change their state and the BIOS is shown on the display.

If there is no image on the display, press `Scroll Lock` (`Caps Shift+Symbol Shift+G` on +UNO) to switch between RGB and VGA modes, just in case the recovery BIOS did start in the wrong mode for our setup.




10. Insert in the ZXUNO+ the SD card formatted as FAT16 o FAT32, and with the `FLASH.ZX1` file downloaded previously.
11. Unplug the USB-Blaster.

12. Select the option `Upgrade Flash from SD`. Press Enter, choose `Yes`, and press Enter again to start the Flash writing process.


This process can't be undone, and it will replace all the previously installed cores, the BIOS, the ZX Spectrum ROMs and their configuration with the data in the image file.

Usually, the recovery image is set to use a PS/2 keyboard so, for a +UNO, some key combinations, like `Caps Shift + 5`, etc may not work. In this case, you have to change the keyboard mode to `PC XT`(`Caps Shift + Symbol Shift + U` and then `9`), to make them work temporarily.

13. After some minutes, the process will end, and, after turning the ZXUNO+ off and on, it should start fine.

If no image is shown, press again `Scroll Lock` (`Caps Shift+Symbol Shift+G` on +UNO) to switch between RGB and VGA modes. In this case, you should have to enter the BIOS and change the right advanced setting that matches your display.

For a +UNO, since the recovery image uses a PS/2 configuration as default, follow this steps to set up the BIOS correctly:

1. If you see no image, switch between RGB and VGA mode (`Caps Shift+Symbol Shift+G`)
2. Change to `PC XT` keyboard mode (`Caps Shift + Symbol Shift + U` and then `9`)
3. Reboot +UNO without losing the temporary keyboard mode (`Caps Shift + Symbol Shift + B`)
4. Quickly, press `Caps Shift + 1`
5. Again, if there's no image, switch between RGB and VGA mode (`Caps`

> Shift+Symbol Shift+G)
>
> 6. Navigate through BIOS and turn on these options:
>    - Advanced → Keyboard Layout: Spectrum
>    - Advanced → Video: VGA (only if there was no image)
> 7. Save changes:
>    - Exit → Save changes and exit
> 8. Completely turn off the +UNO and turn it on again

# References

## Scan Codes

101-, 102-, and 104-key Scan Codes

| KEY | MAKE | BREAK |
|---|---|---|
| A | 1C | F0,1C |
| B | 32 | F0,32 |
| C | 21 | F0,21 |
| D | 23 | F0,23 |
| E | 24 | F0,24 |
| F | 2B | F0,2B |
| G | 34 | F0,34 |
| H | 33 | F0,33 |
| I | 43 | F0,43 |
| J | 3B | F0,3B |
| K | 42 | F0,42 |
| L | 4B | F0,4B |
| M | 3A | F0,3A |
| N | 31 | F0,31 |
| O | 44 | F0,44 |
| P | 4D | F0,4D |
| Q | 15 | F0,15 |
| R | 2D | F0,2D |
| S | 1B | F0,1B |
| T | 2C | F0,2C |
| U | 3C | F0,3C |
| V | 2A | F0,2A |
| W | 1D | F0,1D |
| X | 22 | F0,22 |
| Y | 35 | F0,35 |
| Z | 1A | F0,1A |
| 0 | 45 | F0,45 |
| 1 | 16 | F0,16 |
| 2 | 1E | F0,1E |
| 3 | 26 | F0,26 |
| 4 | 25 | F0,25 |
| 5 | 2E | F0,2E |
| 6 | 36 | F0,36 |
| 7 | 3D | F0,3D |
| 8 | 3E | F0,3E |
| 9 | 46 | F0,46 |
| ` | 0E | F0,0E |
| - | 4E | F0,4E |
| = | 55 | FO,55 |
| \ | 5D | F0,5D |
| BKSP | 66 | F0,66 |
| SPACE | 29 | F0,29 |
| TAB | 0D | F0,0D |
| CAPS | 58 | F0,58 |
| L SHFT | 12 | FO,12 |
| L CTRL | 14 | FO,14 |
| L GUI | E0,1F | E0,F0,1F |
| L ALT | 11 | F0,11 |
| R SHFT | 59 | F0,59 |
| R CTRL | E0,14 | E0,F0,14 |
| R GUI | E0,27 | E0,F0,27 |
| R ALT | E0,11 | E0,F0,11 |
| APPS | E0,2F | E0,F0,2F |
| ENTER | 5A | F0,5A |
| ESC | 76 | F0,76 |
| F1 | 5 | F0,05 |
| F2 | 6 | F0,06 |
| F3 | 4 | F0,04 |
| F4 | 0C | F0,0C |
| F5 | 3 | F0,03 |
| F6 | 0B | F0,0B |
| F7 | 83 | F0,83 |
| F8 | 0A | F0,0A |
| F9 | 1 | F0,01 |
| F10 | 9 | F0,09 |
| F11 | 78 | F0,78 |
| F12 | 7 | F0,07 |
| PRNT SCRN | E0,12, E0,7C | E0,F0, 7C,E0, F0,12 |
| SCROLL | 7E | F0,7E |
| PAUSE | E1,14,77, E1,F0,14,F0,77 | -NONE- |

101-, 102-, and 104-key Scan Codes

| KEY | MAKE | BREAK |
|---|---|---|
| [ | 54 | FO,54 |
| INSERT | E0,70 | E0,F0,70 |
| HOME | E0,6C | E0,F0,6C |
| PG UP | E0,7D | E0,F0,7D |
| DELETE | E0,71 | E0,F0,71 |
| END | E0,69 | E0,F0,69 |
| PG DN | E0,7A | E0,F0,7A |
| U ARROW | E0,75 | E0,F0,75 |
| L ARROW | E0,6B | E0,F0,6B |
| D ARROW | E0,72 | E0,F0,72 |
| R ARROW | E0,74 | E0,F0,74 |
| NUM | 77 | F0,77 |
| KP / | E0,4A | E0,F0,4A |
| KP * | 7C | F0,7C |
| KP - | 7B | F0,7B |
| KP + | 79 | F0,79 |
| KP EN | E0,5A | E0,F0,5A |
| KP . | 71 | F0,71 |
| KP 0 | 70 | F0,70 |
| KP 1 | 69 | F0,69 |
| KP 2 | 72 | F0,72 |
| KP 3 | 7A | F0,7A |
| KP 4 | 6B | F0,6B |
| KP 5 | 73 | F0,73 |
| KP 6 | 74 | F0,74 |
| KP 7 | 6C | F0,6C |
| KP 8 | 75 | F0,75 |
| KP 9 | 7D | F0,7D |
| ] | 5B | F0,5B |
| ; | 4C | F0,4C |
| ' | 52 | F0,52 |
| , | 41 | F0,41 |
| . | 49 | F0,49 |
| / | 4A | F0,4A |

Windows Multimedia Scan Codes

| Key | Make Code | Break Code |
|---|---|---|
| Next Track | E0, 4D | E0, F0, 4D |
| Previous Track | E0, 15 | E0, F0, 15 |
| Stop | E0, 3B | E0, F0, 3B |
| Play/Pause | E0, 34 | E0, F0, 34 |
| Mute | E0, 23 | E0, F0, 23 |
| Volume Up | E0, 32 | E0, F0, 32 |
| Volume Down | E0, 21 | E0, F0, 21 |
| Media Select | E0, 50 | E0, F0, 50 |
| E-Mail | E0, 48 | E0, F0, 48 |
| Calculator | E0, 2B | E0, F0, 2B |
| My Computer | E0, 40 | E0, F0, 40 |
| WWW Search | E0, 10 | E0, F0, 10 |
| WWW Home | E0, 3A | E0, F0, 3A |
| WWW Back | E0, 38 | E0, F0, 38 |
| WWW Forward | E0, 30 | E0, F0, 30 |
| WWW Stop | E0, 28 | E0, F0, 28 |
| WWW Refresh | E0, 20 | E0, F0, 20 |
| WWW Favorites | E0, 18 | E0, F0, 18 |

ACPI Scan Codes

| Key | Make Code | Break Code |
|---|---|---|
| Power | E0, 37 | E0, F0, 37 |
| Sleep | E0, 3F | E0, F0, 3F |
| Wake | E0, 5E | E0, F0, 5E |

# References

# Spectrum

## Scan Codes

101-, 102-, and 104-key Scan Codes

| KEY | MAKE | BREAK |
|---|---|---|
| A | 1C | F0,1C |
| B | 32 | F0,32 |
| C | 21 | F0,21 |
| D | 23 | F0,23 |
| E | 24 | F0,24 |
| F | 2B | F0,2B |
| G | 34 | F0,34 |
| H | 33 | F0,33 |
| I | 43 | F0,43 |
| J | 3B | F0,3B |
| K | 42 | F0,42 |
| L | 4B | F0,4B |
| M | 3A | F0,3A |
| N | 31 | F0,31 |
| O | 44 | F0,44 |
| P | 4D | F0,4D |
| Q | 15 | F0,15 |
| R | 2D | F0,2D |
| S | 1B | F0,1B |
| T | 2C | F0,2C |
| U | 3C | F0,3C |
| V | 2A | F0,2A |
| W | 1D | F0,1D |
| X | 22 | F0,22 |
| Y | 35 | F0,35 |
| Z | 1A | F0,1A |
| 0 | 45 | F0,45 |
| 1 | 16 | F0,16 |
| 2 | 1E | F0,1E |
| 3 | 26 | F0,26 |
| 4 | 25 | F0,25 |
| 5 | 2E | F0,2E |
| 6 | 36 | F0,36 |
| 7 | 3D | F0,3D |
| 8 | 3E | F0,3E |
| 9 | 46 | F0,46 |
| ` | 0E | F0,0E |
| - | 4E | F0,4E |
| = | 55 | F0,55 |
| \ | 5D | F0,5D |
| BKSP | 66 | F0,66 |
| SPACE | 29 | F0,29 |
| TAB | 0D | F0,0D |
| CAPS | 58 | F0,58 |
| L SHFT | 12 | F0,12 |
| L CTRL | 14 | F0,14 |
| L GUI | E0,1F | E0,F0,1F |
| L ALT | 11 | F0,11 |
| R SHFT | 59 | F0,59 |
| R CTRL | E0,14 | E0,F0,14 |
| R GUI | E0,27 | E0,F0,27 |
| R ALT | E0,11 | E0,F0,11 |
| APPS | E0,2F | E0,F0,2F |
| ENTER | 5A | F0,5A |
| ESC | 76 | F0,76 |
| F1 | 5 | F0,05 |
| F2 | 6 | F0,06 |
| F3 | 4 | F0,04 |
| F4 | 0C | F0,0C |
| F5 | 3 | F0,03 |
| F6 | 0B | F0,0B |
| F7 | 83 | F0,83 |
| F8 | 0A | F0,0A |
| F9 | 1 | F0,01 |
| F10 | 9 | F0,09 |
| F11 | 78 | F0,78 |
| F12 | 7 | F0,07 |
| PRNT SCRN | E0,12, E0,7C | E0,F0, 7C,E0, F0,12 |
| SCROLL | 7E | F0,7E |
| PAUSE | E1,14,77, E1,F0,14,F0,77 | -NONE- |

101-, 102-, and 104-key Scan Codes

| KEY | MAKE | BREAK |
|---|---|---|
| [ | 54 | FO,54 |
| INSERT | E0,70 | E0,F0,70 |
| HOME | E0,6C | E0,F0,6C |
| PG UP | E0,7D | E0,F0,7D |
| DELETE | E0,71 | E0,F0,71 |
| END | E0,69 | E0,F0,69 |
| PG DN | E0,7A | E0,F0,7A |
| U ARROW | E0,75 | E0,F0,75 |
| L ARROW | E0,6B | E0,F0,6B |
| D ARROW | E0,72 | E0,F0,72 |
| R ARROW | E0,74 | E0,F0,74 |
| NUM | 77 | F0,77 |
| KP / | E0,4A | E0,F0,4A |
| KP * | 7C | F0,7C |
| KP - | 7B | F0,7B |
| KP + | 79 | F0,79 |
| KP EN | E0,5A | E0,F0,5A |
| KP . | 71 | F0,71 |
| KP 0 | 70 | F0,70 |
| KP 1 | 69 | F0,69 |
| KP 2 | 72 | F0,72 |
| KP 3 | 7A | F0,7A |
| KP 4 | 6B | F0,6B |
| KP 5 | 73 | F0,73 |
| KP 6 | 74 | F0,74 |
| KP 7 | 6C | F0,6C |
| KP 8 | 75 | F0,75 |
| KP 9 | 7D | F0,7D |
| ] | 5B | F0,5B |
| ; | 4C | F0,4C |
| | 52 | F0,52 |
| , | 41 | F0,41 |
| . | 49 | F0,49 |
| / | 4A | F0,4A |

Windows Multimedia Scan Codes

| Key | Make Code | Break Code |
|---|---|---|
| Next Track | E0, 4D | E0, F0, 4D |
| Previous Track | E0, 15 | E0, F0, 15 |
| Stop | E0, 3B | E0, F0, 3B |
| Play/Pause | E0, 34 | E0, F0, 34 |
| Mute | E0, 23 | E0, F0, 23 |
| Volume Up | E0, 32 | E0, F0, 32 |
| Volume Down | E0, 21 | E0, F0, 21 |
| Media Select | E0, 50 | E0, F0, 50 |
| E-Mail | E0, 48 | E0, F0, 48 |
| Calculator | E0, 2B | E0, F0, 2B |
| My Computer | E0, 40 | E0, F0, 40 |
| WWW Search | E0, 10 | E0, F0, 10 |
| WWW Home | E0, 3A | E0, F0, 3A |
| WWW Back | E0, 38 | E0, F0, 38 |
| WWW Forward | E0, 30 | E0, F0, 30 |
| WWW Stop | E0, 28 | E0, F0, 28 |
| WWW Refresh | E0, 20 | E0, F0, 20 |
| WWW Favorites | E0, 18 | E0, F0, 18 |

ACPI Scan Codes

| Key | Make Code | Break Code |
|---|---|---|
| Power | E0, 37 | E0, F0, 37 |
| Sleep | E0, 3F | E0, F0, 3F |
| Wake | E0, 5E | E0, F0, 5E |

## ZX-Uno control I/O registers

In the Spectrum core the $FC3B and $FD3B ports are reserved and assigned by the ZXI committee. A total of 256 different I/O registers unique to the ZX-Uno family are accessed through these ports.

Port $FC3B (64571) stores the address ($00 - $FF) of the I/O register to be accessed. It can be read to find out which register address was last allocated.

Port $FD3B (64827) is the access port to the register selected with the previous port. Its meaning (read/write) depends on the implementation of each register.

For example, to allocate bank 16 of the SRAM to the address space $C000 - $FFFFFF during boot mode, using the MASTERMAPPER register, would be as follows:

```
ld bc,$fc3b     ;Port to set register number to use
ld a,1          ;Register $01 (MASTERMAPPER)
out (c),a       ;Selected. From now on, any access to $FD3B is using MASTERMAPPER
inc b           ;Register access port ($FD3B, just increment B)
ld a,16         ;SRAM Bank 16
out (c),a       ;Write to the MASTERMAPPER register
```

The registers implemented in the ZX-Uno family are described below.

### $00 MASTERCONF

| Direction | Value after user reset | Value after master reset | Value after poweron |
|---|---|---|---|
| Read/Write | Does not change | 00000001 | 00000001 |

Binary format (fields in **bold** can only be altered when LOCK=0):

| **LOCK** | MODE1 | DISCONT | MODE0 | I2KB | **DISNMI** | **DIVEN** | **BOOTM** |
|---|---|---|---|---|---|---|---|

- BOOTM: 1 indicates that the ZX-Uno is in boot mode (configuration mode). Boot mode only makes sense while the boot firmware is running, where some aspects of the ZX-Uno are allowed to be configured before going into run mode. MASTERCONF can always be read, both in boot mode and in run mode. It is set to 0 manually per program, at which point ZX-Uno enters run mode.
- DIVEN: set to 0 indicates that DIVMMC is not enabled in the system, although the SPI interface access ports of the SD/MMC slot are still available. The memory used by DIVMMC remains available for other uses. 1 indicates that DIVMMC is enabled. If enabled, an ESXDOS image must be loaded into the corresponding RAM bank before entering run mode. The default value of this bit is 0.
- DISNMI: set to 1 indicates that the DIVMMC NMI function will not be available. NMI will work, but will not cause ESXDOS to automate, thus leaving NMI control to the main system ROM. Bit added to improve DIVMMC compatibility with SE Basic IV. Defaults to 0 (ESXDOS handles NMI events).
- I2KB: a1 sets the ULA to return a value consistent with a Spectrum issue 2 when reading the keyboard port ($FE). When 0, the returned value is compatible with issue 3 and later. It defaults to 0.
- MODE1,MODE0: specifies the ULA timing mode to accommodate different Spectrum models. 00 = ULA ZX Spectrum 48K PAL, 01 = ZX Spectrum 128K/+2 grey, 10 = Pentagon 128, 11 = 48K NTSC (262 scans).
- DISCONT: indicates whether memory contention should occur in the video memory. 0 to enable contention (48K and 128K compatibility). 1 to disable contention (Pentagon 128 compatibility).
- LOCK: When set to 1, it prevents further changes to certain bits in the MASTERCONF register, and also prevents access to the SPI Flash. This bit is reset to 0 only by a master reset (Ctrl-Alt-BkSpace) or by powering the clone off and on.

### $01 MASTERMAPPER

| Direction | Value after user reset | Value after master reset | Value after poweron |
|---|---|---|---|
| Read/Write | Does not change | 00000000 | 00000000 |

Only the lower 5 bits (values $00 to $1F) of this register are used. The value stored is the number of a 16KB bank of SRAM that will be paged at addresses $C000-$FFFFFF during boot mode. The values in this register have no effect when the ZX-Uno is in run mode. 32 different values for this register allow addressing up to 512KB of SRAM. If ZX-Uno is expanded with more memory, more bits in this register will be used. The maximum manageable amount of memory is 4MB.

### $02 FLASHSPI

| Direction | Value after user reset | Value after master reset | Value after poweron |
|---|---|---|---|
| Read/Write | Does not change | Does not change | 00000000 |

Puerto de acceso al registro SPI conectado a la SPI Flash. Escribiendo un valor en este registro, se envía a la SPI Flash, si ésta está seleccionada. Leyendo un valor de este registro, se lee el último valor enviado por la SPI Flash, y además, la misma operación de lectura provoca que la SPI envíe un nuevo valor (que sería leído con la siguiente operación de lectura a este registro). Por esta razón, en operaciones de lectura de bloques, el primer byte leído con este puerto debe descartarse.

### $03 FLASHCS

| Direction | Value after user reset | Value after master reset | Value after poweron |
|---|---|---|---|
| Read/Write | Does not change | Does not change | 00000001 |

Only bit 0 is used. The value written to this register determines the status of the CS line of the SPI Flash (0 = Flash selected, 1 = Flash not selected).

### $04 SCANCODE

| Direction | Value after user reset | Value after master reset | Value after poweron |
|---|---|---|---|
| Read/Write | Does not change | Does not change | Does not change |

When reading, it allows to obtain the value of the last scancode generated by the keyboard. When writing, it allows to send commands to the keyboard.

### $05 KEYSTAT

| Direction | Value after user reset | Value after master reset | Value after poweron |
|---|---|---|---|
| Read | Does not change | Does not change | Does not change |

Several bits showing whether or not there is a new key pressed, or released, and whether this is an extended or normal key.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| BSY | 0 | 0 | 0 | ERR | RLS | EXT | PEN |

- BSY: Uses 1 when a data transmission to the PS/2 port is still in progress. Wait for a value of 0 to start a new transmission.
- ERR: 1 when the last transmission to or from the PS/2 port had errors.
- RLS: 1 when the last event belongs to a key that has been released.
- EXT: 1 when the last event belongs to a key with extended code (E0+scancode).
- PEN: 1 when there is new data ready to be read in the SCANCODE register. After reading KEYSTAT, this bit is set to 0.

### $06 JOYCONF

| Direction | Value after user reset | Value after master reset | Value after poweron |
|---|---|---|---|
| Read/Write | Does not change | Does not change | 00100001 |

Bits 0 to 2 show the operating mode of the keyboard mapped joystick (or of the second physical joystick if there is a splitter). Bits 4 to 6 for the operating mode of the physical joystick (DB-9 connector on the side). Bit 3 means autofire of the second joystick or keypad. Bit 7 means autofire of the main joystick. The values are: 000 = Disabled, 001 = Kempston, 010 = Sinclair 1, 011 = Sinclair 2, 100 = Protek/Cursor/AGF, 101 = Fuller, 110 = OPQAspM, 111 = reserved.

### $07 KEYMAP

| Direction | Value after user reset | Value after master reset | Value after poweron |
|---|---|---|---|
| Read/Write | Read later | Read later | Read later |

On read, each access provides the next byte of the currently loaded ZX-Uno keymap. On write, the byte corresponding to the keymap marked by the current position is modified. In both cases, the address pointer is automatically incremented to point to the next byte of the keymap. This pointer returns to 0 automatically after a reset, a write to the $FC3B register, or when the keymap is terminated. The keymap, in the current implementation, occupies 16384 bytes.

### $09 MOUSEDATA

| Direction | Value after user reset | Value after master reset | Value after poweron |
|---|---|---|---|
| Read/Write | Does not change | Does not change | Does not change |

Mouse PS/2 port data register. Used to read or send direct commands to the PS/2 mouse. For example: to initialise the mouse, the value $F4 must be sent to this register.

### $0A MOUSESTATUS

| Direction | Value after user reset | Value after master reset | Value after poweron |
|---|---|---|---|
| Read/Write | Does not change | Does not change | Does not change |

PS/2 mouse port status register. The following bits are set:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| BSY | 0 | 0 | 0 | ERR | 0 | 0 | PEN |

- BSY: set to 1 when a data transmission to the PS/2 port is still in progress. Wait for a value of 0 to start a new transmission.
- ERR: set to 1 when the last transmission to or from the PS/2 port had errors.
- PEN: set to 1 when new data is ready to be read from the MOUSEDATA register. After reading MOUSESTATUS, this bit is set to 0.

### $0B SCANDBLCTRL

| Direction | Value after user reset | Value after master reset | Value after poweron |
|---|---|---|---|
| Read/Write | Does not change | Does not change | 00000000 |

Scandoubler control register and system speed. The following bits are defined:

| | | | | | |
|---|---|---|---|---|---|
| TURBO | COPT | FREQ | FREQ | ENSCAN | VGA |

- TURBO: 00 to select 3.5 MHz, 01 to select 7 MHz, 10 to select 14 MHz and 11 to select 28 MHz. These bits are also updated with the value of bits D0-D3 of port $8E3B, used in the ZX Prism to select the different speeds for the CPU.
- COPT: selects the way to generate the composite syncs for RGB and composite video: 0 to use the original Spectrum sync type. 1 to use sync pulses according to the PAL standard.
- FREQ: these three bits define the frequency of the master clock, from which all other clocks in the circuit are derived. Among others, the vertical refresh rate is also defined here, which can be used to improve compatibility with some VGA monitors that do not support a vertical refresh rate of 50Hz. The vertical refresh rate values are as follows:
  - 000 : 50Hz for 48K and Pentagon mode.
  - 001 : 50Hz for 128K mode
  - 010 : 52 Hz
  - 011 : 53 Hz
  - 100 : 55 Hz
  - 101 : 57 Hz
  - 110 : 59 Hz
  - 111 : 60 Hz
- ENSCAN: to 1 to enable the scanline effect in VGA mode. No effect if VGA mode is disabled.
- VGA: set to 1 to enable the scandoubler. The scandoubler output is the same as the normal RGB output, but with a doubling of the horizontal delay frequency. Set to 0 to use 15kHz RGB / composite video output.

### $0C RASTERLINE

| Direction | Value after user reset | Value after master reset | Value after poweron |
|---|---|---|---|
| Read/Write | 11111111 | 11111111 | 11111111 |

Stores the least significant 8 bits of the screen line where maskable interrupt should be triggered. A value of 0 for this register (with LINE8 also equal to 0) sets the raster interrupt to be triggered, if enabled, right at the start of the right edge of the line before the first screen line where the "paper" area begins. In other words: the line count of this interrupt assumes that a screen line is composed of: right edge + horizontal blanking interval + left edge + paper zone. If this assumption is made, the interrupt would be triggered at the beginning of the selected line. A value for RASTERLINE equal to 192 (with LINE8 equal to 0) triggers the raster interrupt at the beginning of the bottom edge. The line numbers for the end of the bottom edge and start of the top edge depend on the timings used. The highest value possible in practice for RASTERLINE corresponds to a raster interrupt triggered at the last line of the top edge (see RASTERCTRL).

### $0D RASTERCTRL

| Direction | Value after user reset | Value after master reset | Value after poweron |
|---|---|---|---|
| Read/Write | 00000001 | 00000001 | 00000001 |

Raster Interrupt Status and Control Register. The following bits are defined.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| INT | 0 | 0 | 0 | 0 | 0 | DISVINT | ENARINT | LINE8 |

- INT: this bit is only available on read. It is set to 1 for 32 clock cycles from the time the raster interrupt is triggered. This bit is available even if the processor has interrupts disabled. It is not available if the ENARINT bit is set to 0.
- DISVINT: set to 1 to disable the vertical retrace maskable interrupts (the original ULA interrupts). After a reset, this bit is set to 0.
- ENARINT: set to 1 to enable raster line maskable interrupts. After a reset, this bit is set to 0.
- LINE8: saves bit 8 of the RASTERLINE value, so that any value between 0 and 511 can be set, although in practice, the largest value is limited by the number of lines generated by the ULA (311 in 48K mode, 310 in 128K mode, 319 in Pentagon mode). If a line number higher than the limit is set, the raster interrupt will not occur.

### $0E DEVCONTROL

| Direction | Value after user reset | Value after master reset | Value after poweron |
|---|---|---|---|
| Read/Write | Does not change | 00000000 | 00000000 |

Enable/disable register for different features. The following bits are defined.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| DISD | ENMMU | DIROMSEL1F | DIROMSEL7F | DI1FFD | DI7FFD | DI7FFD | DITAY | DIAY |

- DISD: Set to 1 to disable the SPI hardware interface for SD (used in DivMMC and ZXMMC). Disabling this interface frees ports $1F on write, $3F, $E7 and $EB. After master reset, it is set to 0.
- ENMMU: set to 1 to enable the horizontal MMU used on the Timex Sinclair. Enabling this interface uses bit 7 of port $FF, port $F4 is used and a read to port $FF returns the last value written to it. After master reset, it is 0.
- DIROM1F: the value of this bit is masked with the value of bit 2 of port $1FFD according to the operation ~DIROM1F & $1FFD[2]. The net result is that if this bit is set to 1, the system will work as if the value of bit 2 of port $1FFD is always 0, regardless of the value written to it. After master reset, it is set to 0, which allows changes to bit 2 of $1FFD to be taken into account.
- DIROM7F: the value of this bit is masked with the value of bit 4 of port $7FFD according to the operation ~DIROM7F & $7FFD[4]. The net result is that if this bit is set to 1, the system will work as if the value of bit 4 of port $7FFD is always 0, regardless of the value written to it. After master reset, it is set to 0, which allows changes to bit 4 of $7FFD to be taken into account.
- DI1FFD: set to 1 to disable the +2A/+3 compatible paging system. Disabling this interface frees the $1FFD port on write. Note that the decoding of port $7FFD, if active, is different depending on whether port $1FFD is active or not.
- DI7FFD: to 1 to disable the 128K compatible paging system. Disabling this system disables the +2A/+3 paging system even if it is not explicitly disabled.
- DITAY: set to 1 to disable the second AY chip, thus disabling Turbo Sound mode.
- DIAY: set to 1 to disable the main AY chip. Disabling this chip disables the second AY chip, even if it is not explicitly disabled.

### $0F DEVCTRL2

| Direction | Value after user reset | Value after master reset | Value after poweron |
|---|---|---|---|
| Read/Write | Does not change | 00000000 | 00000000 |

|Enable/disable register of different features (continuation of DEVCONTROL). The following bits are defined.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Resv | Resv | SPLITTER | DIMIXER | DISPECDRUM | DIRADES | DITIMEX | DIULAPLUS |

- Resv: This bit is reserved. In the current implementation, a 0 must be written to it if the register value is updated.
- SPLITTER: set to 1 to enable joystick splitter.
- DIMIXER: set to 1 to disable sound output.
- DISPECDRUM: set to 1 to disable SpecDrum/Covox support.
- DIRADAS: set to 1 to disable radastanian mode. Note that if radastanian mode is not disabled, but ULAplus is disabled, when attempting to use radastanian mode, the datapath used in the ULA will not be as expected and the display behaviour in this case is not documented.
- DITIMEX: to 1 to disable Timex compatible display modes. Any writes to the $FF port are therefore ignored. If the Timex MMU is enabled, a read to port $FF will return 0.
- DIULAPLUS: set to 1 to disable the ULAplus. Any writes to the ULAplus ports are ignored. Reads to those ports return the value of the floating bus. Note however that the contention mechanism for this port still works even if it is disabled.

### $10 MEMREPORT

| Direction | Value after user reset | Value after master reset | Value after poweron |
|---|---|---|---|
| Read/Write | Does not change | 00000000 | 00000000 |

### $40 RADASCTRL

| Direction | Value after user reset | Value after master reset | Value after poweron |
|---|---|---|---|
| Read/Write | 00000000 | 00000000 | 00000000 |

Register for setting the radastanian mode and its characteristics. Writes to this register are ignored if the corresponding bit in DEVCTRL2 is set. The following bits are defined

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Resv | Resv | Resv | Resv | Resv | Resv | Resv | Resv | EN1 | EN0 |

- Resv: This bit is reserved. In the current implementation, a 0 must be written to it if the register value is updated.
- EN1, EN0: Both bits must be set to 1 to enable Radastanian mode. If EN0 is enabled, but EN1 is disabled, the rest of the bits, from 2 to 7, are defined as implemented in the ZEsarUX emulator (see César Hernández's documentation on this).

### $41 RADASOFFSET

| Direction | Value after user reset | Value after master reset | Value after poweron |
|---|---|---|---|
| Read/Write | 00000000 | 00000000 | 00000000 |

Contains the number of bytes to be added to the base address of the screen (4000h, 6000h, C000h or E000h depending on the configuration) to obtain the address where the first two pixels are located in Radastanian mode. It is a 14-bit register. To write a value, the 8 least significant bits are written first, followed immediately by the 8 most significant bits (the 2 most significant bits of this value are ignored). If the offset value is such that scanning the screen memory to create the image reaches the end of a 16KB page, scanning will continue at the beginning of that same page.

### $42 RADASPADDING

| Direction | Value after user reset | Value after master reset | Value after poweron |
|---|---|---|---|
| Read/Write | 00000000 | 00000000 | 00000000 |

Contains the number of bytes - 64 that a scanline occupies in Radastanian mode. That is, if this register is 0, the length in bytes of a scanline is 64 bytes (128 pixels). If this register is 4, the length of a scanline is 68 bytes (136 pixels). If it is 255, the length of a scanline is 64+255=319 bytes, or 638 pixels. If the offset value is such that scanning the screen memory to create the image reaches the end of a 16KB page, scanning will continue at the beginning of that same page.

### $43 RADASPALBANK

| Direction | Value after user reset | Value after master reset | Value after poweron |
|---|---|---|---|
| Read/Write | 00000000 | 00000000 | 00000000 |

Register used to set which section of the ULAplus palette will define the colours in Radastanian mode, and how the border will behave.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Resv | Resv | Resv | Resv | Resv | Resv | BOR3 | RADPALQUARTER |

- Resv: This bit is reserved. In the current implementation, a 0 must be written to it if the register value is updated.
- BOR3: Within the currently selected palette, indicates whether the border colour will be taken from entries 0 to 7 (0) or from entries 8 to 15 (1). It can be considered as bit 3 of the border colour in Radastanian mode.
- RADPALQUARTER: two bits indicating which section of the ULAplus palette is to be used for the Radastanian mode. 00 for using the section of inputs 0 to 15. 01 for inputs 16 to 31, 10 for inputs 32 to 47 and 11 for using inputs 48 to 63.

### $80 HOFFS48K

| Direction | Value after user reset | Value after master reset | Value after poweron |
|---|---|---|---|
| Read/Write | Does not change | Does not change | $38 |

Horizontal screen centering adjustment value for 48K ULA.

### $81 VOFFS48K

| Direction | Value after user reset | Value after master reset | Value after poweron |
|---|---|---|---|
| Read/Write | Does not change | Does not change | $01 |

Vertical screen centering adjustment value for 48K ULA.

### $82 HOFFS128K

| Direction | Value after user reset | Value after master reset | Value after poweron |
|---|---|---|---|
| Read/Write | Does not change | Does not change | $3A |

Horizontal screen centering adjustment value for 128K ULA.

### $83 VOFFS128K

| Direction | Value after user reset | Value after master reset | Value after poweron |
|---|---|---|---|
| Read/Write | Does not change | Does not change | $01 |

Vertical screen centering adjustment value for 128K ULA.

### $84

| Direction | Value after user reset | Value after master reset | Value after poweron |
|---|---|---|---|
| HOFFSPEN | Read/Write | Does not change | Does not change |

Horizontal screen centering adjustment value for Pentagon ULA.

### $85 VOFFSPEN

| Direction | Value after user reset | Value after master reset | Value after poweron |
|---|---|---|---|
| Read/Write | Does not change | Does not change | $00 |

Vertical screen centering adjustment value for Pentagon ULA.

### $A0 DMACTRL

| Direction | Value after user reset | Value after master reset | Value after poweron |
|---|---|---|---|
| Read/Write | Does not change | 00000000 | 00000000 |

### $A1 DMASRC

| Direction | Value after user reset | Value after master reset | Value after poweron |
|---|---|---|---|
| Read/Write | Does not change | 00000000 | 00000000 |

### $A2 DMADST

| Direction | Value after user reset | Value after master reset | Value after poweron |
|---|---|---|---|
| Read/Write | Does not change | 00000000 | 00000000 |

### $A3 DMAPRE

| Direction | Value after user reset | Value after master reset | Value after poweron |
|---|---|---|---|
| Read/Write | Does not change | 00000000 | 00000000 |

### $A4 DMALEN

| Direction | Value after user reset | Value after master reset | Value after poweron |
|---|---|---|---|
| Read/Write | Does not change | 00000000 | 00000000 |

### $A5 DMAPROB

| Direction | Value after user reset | Value after master reset | Value after poweron |
|---|---|---|---|
| Read/Write | Does not change | 00000000 | 00000000 |

### $A6 DMASTAT

| Direction | Value after user reset | Value after master reset | Value after poweron |
|---|---|---|---|
| Read/Write | Does not change | 00000000 | 00000000 |

### $C6 UARTDATA

| Direction | Value after user reset | Value after master reset | Value after poweron |
|---|---|---|---|
| Read/Write | Does not change | 00000000 | 00000000 |

### $C7 UARTSTAT

| Direction | Value after user reset | Value after master reset | Value after poweron |
|---|---|---|---|
| Read/Write | Does not change | 00000000 | 00000000 |

### $C8 - $DF RESERVED

| Direction | Value after user reset | Value after master reset | Value after poweron |
|---|---|---|---|
| Read/Write | Definido por el usuario | Definido por el usuario | Definido por el usuario |

ZXUNO registers reserved for experiments or private use.

### $F0 SRAMADDR

| Direction | Value after user reset | Value after master reset | Value after poweron |
|---|---|---|---|
| Read/Write | Does not change | 00000000 | 00000000 |

### $F1 MADDRINC

| Direction | Value after user reset | Value after master reset | Value after poweron |
|---|---|---|---|
| Read/Write | Does not change | 00000000 | 00000000<br><br>===== $F2 SRAMDATA<br><br>[align="center",options="header"] |

|Direction|Value after user reset|Value after master reset|Value after poweron
|Lectura/Escritura|Does not change|00000000|00000000

### $F3 VDECKCTRL

| Direction | Value after user reset | Value after master reset | Value after poweron |
|---|---|---|---|
| Read/Write | Does not change | 00000000 | 00000000 |

### $F7 AUDIOMIX

| Direction | Value after user reset | Value after master reset | Value after poweron |
|---|---|---|---|
| Read/Write | Does not change | Does not change | 10011111 |

Control of the left/right channel mix from the 4 channels of the AY-8912. The following groups of 2 bits each are defined:

| === | CHANNELA | CHANNELB | CHANNELC | BEEPDRUM |
|---|---|---|---|---|

- BEEPDRUM : Channel for Beeper and Specdrum.
- The meaning of each group of 2 bits is: 00 = Mute, 01 = Right channel, 10 = Left channel, 11 = Both.

### $FB AD724

| Direction | Value after user reset | Value after master reset | Value after poweron |
|---|---|---|---|
| Read/Write | Does not change | Does not change | 00000000 |

Control of the operating mode of the AD724 encoder chip. The following bits are defined:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Resv | Resv | Resv | Resv | Resv | Resv | Resv | Resv | Resv | MODE |

- Resv : the meaning of these bits is reserved and must not be altered.
- MODE : operating mode of the AD724. 0 = encodes PAL standard. 1 = encodes NTSC standard.

### $FC COREADDR

| Direction | Value after user reset | Value after master reset | Value after poweron |
|---|---|---|---|
| Read/Write | Does not change | Does not change | $058000 |

Stores the address, within SPI memory, of the start of the core to be booted. To store a different address, three writes must be made to this register, containing the three bytes of the address, in order from most significant to least significant. On read, each access to this register returns a part of the last stored address, from the most significant part to the least significant part.

### $FD COREBOOT

| Direction | Value after user reset | Value after master reset | Value after poweron |
|---|---|---|---|
| Read | Does not change | Does not change | Does not change |

Boot control register. Writing a 1 to bit 0 of this register (all other bits are reserved and must remain at 0) triggers the FPGA's internal mechanism to start another core. The start address of this second core will be the last one written using the COREADDR register.

### $FE SCRATCH

| Direction | Value after user reset | Value after master reset | Value after poweron |
|---|---|---|---|
| Read/Write | Does not change | 00000000 | 00000000 |

### $FF COREID

| Direction | Value after user reset | Value after master reset | Value after poweron |
|---|---|---|---|
| Read | Read later | Read later | Read later |

Each read operation provides the next ASCII character of the string containing the current revision of the ZX-Uno core. When the string ends, subsequent reads output bytes with the value 0 (at least on) until the string starts again. This pointer returns to 0 automatically after a reset or a write to the $FC3B register. The delivered characters that are part of the string are standard printable ASCII (codes 32-127). Any other value indicates that this register is not operational.

# Links

ZX-Uno

foroFPGA

canal oficial de Telegram de ZX-Uno

ZX-Uno FAQ

Guía rápida del ZX-Uno

Wiki de ZX-Uno.

Core ZX Spectrum

The ZX Spectrum +3e Homepage

Sharing a +3e disk with PC (FAT) partitions

Configuración de teclado de ZX-Uno.

Firmware de teclado para ZX Go+

Almost (In-) Complete List of esxDOS DOT-Commands

ESXDOS Manual

WiFi (RetroWiki)

Cargando Leches 2.0

WiFi en ZX-Uno

Core de ZX-Uno Test UART (WiFi)

Network tools for ZX-Uno pack

ESP8266 AT Instruction Set

Vídeos Radastanianos

Nuevo core zx48

ZX 48 for ZX-UNO (Kyp)

Core MSX

MSX1FPGA

MSX Pack

Nextor para MSX

Nextor User Manual

MSX-DOS

Atom Software Archive en carpeta ATOM

Teclado Core Atom

Core de NES para ZX-Uno

ColecoFPGA on GitHub

TOSEC: Magnavox Odyssey 2 (2012-04-23)

Videopac G7000 / Odyssey2 for FPGA

Maxduino - guía de uso

zx123_tool

esxdos

ZX-Modules.

Long Filename Browser de Bob Fossil

New NMI Handler de Dr. Slump

Fuzix

SmartROM

ZX Tape Player

PlayZX

Tapir

Fuse Utilities

MakeTSX

Lynx2Wav

2lynx2wav

cgc2wav

mzf2wav

RetroConverter

Lince Script

Programming a Spartan 6 with a Raspberry Pi

Tutorial para desbriquear el ZX-Uno con una Raspberry

Como programar un UnAmiga con la Raspberry Pi (o Linux) con el USB-Blaster y OpenOCD