



# Manual



# ZXUNO+ Manual

kounch

Version 0.4

# Index

Introduction .....	1
Aknowledgements .....	1
Ports and Connectors .....	2
Description .....	3
RJ-45 Video Out .....	3
Initial Setup .....	4
SD card formatting .....	4
Windows .....	4
MacOS .....	4
Linux .....	5
esxdos .....	6
BIOS .....	8
Main .....	8
ROMs .....	9
Upgrade .....	9
Boot .....	10
Advanced .....	11
Exit .....	12
ZX Spectrum .....	13
SD advanced format (+3e) .....	14
Windows .....	14
MacOS .....	14
Linux .....	16
+3e .....	19
Keyboard .....	20
Spanish .....	20
English .....	20
Spectrum .....	20
Special keys and buttons .....	21
ROMs .....	22
DerbyPro .....	22
CargandoLeches .....	23
POKEs .....	23
Preparing ultrafast loading tapes .....	24
SE Basic IV .....	25
Other ROMs .....	26
esxdos .....	27
Basic Guide .....	27

ZXUNO+ Commands .....	29
Making RDM (RaDastan Movie) files .....	30
Upgrade .....	31
BIOS .....	31
ROMs .....	31
Cores .....	32
esxdos .....	32
Flash Memory .....	33
Other cores .....	34
ZX Spectrum 48K (Kyp) .....	34
SD card format .....	34
Keyboard .....	34
Special keys and buttons .....	34
ZX Spectrum 128K (Kyp) .....	35
SD card format .....	35
Keyboard .....	35
Special keys and buttons .....	35
Acorn Atom .....	36
SD card format .....	36
Keyboard .....	37
Special keys and buttons .....	37
Acorn Electron .....	38
SD card format .....	38
Keyboard .....	38
Special keys and buttons .....	38
Basic Guide .....	39
Basic Guide .....	40
Amstrad CPC 464 .....	41
SD card format .....	41
Keyboard .....	41
Special keys and buttons .....	41
Basic Guide .....	41
Amstrad CPC 6128 .....	42
SD card format .....	42
Keyboard .....	42
Special keys and buttons .....	42
Basic Guide .....	43
Apple II .....	44
SD card formatting .....	44
Windows .....	44
MacOS and Linux .....	44

Keyboard .....	45
Special keys and buttons .....	45
Arcades .....	46
Keyboard .....	46
Special keys and buttons .....	46
Atari 800XL .....	47
SD card .....	47
Keyboard .....	47
Special keys and buttons .....	47
Basic Guide .....	48
Atari 2600 .....	49
SD card format .....	49
Keyboard .....	49
Special keys and buttons .....	49
Basic Guide .....	50
Colecovision .....	51
SD card format .....	51
Keyboard .....	51
Special keys and buttons .....	51
Basic Guide .....	52
Commodore 16 .....	53
SD card format .....	54
Windows .....	54
MacOS and Linux .....	54
Keyboard .....	55
Special keys and buttons .....	55
Basic Guide .....	55
Commodore 64 .....	56
SD card format .....	57
Windows .....	57
MacOS and Linux .....	57
Keyboard .....	58
Special keys and buttons .....	58
Basic Guide .....	59
MSX .....	60
SD card format .....	60
Keyboard .....	61
Special keys and buttons .....	61
Basic Guide .....	62
MSXCTRL .....	62
Other .....	62

NES .....	63
SD card format .....	63
Keyboard .....	63
Special keys and buttons .....	63
Basic Guide .....	64
PC XT .....	65
SD card format .....	65
SmartROM .....	66
SD card format .....	66
ZX81 .....	67
SD card format .....	67
Keyboard .....	67
Basic Guide .....	67
Other Hardware .....	68
Loading from tape .....	68
Cassette Player .....	68
Computer .....	68
PlayTZX .....	68
Mobile phone, tablet, MP3 player, etc. ....	69
Audio file conversion .....	69
Miniduino .....	70
Ports and buttons .....	70
Configuration .....	71
Use .....	72
Maxduino firmware upgrade .....	74
Troubleshooting .....	77
Firmware images management .....	77
zx123_tool .....	77
Firmware recovery .....	81
JTAG cable connections .....	81
Recovery using a Raspberry Pi .....	82
References .....	87

# Introduction

ZXUNO+ is the latest 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 software 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

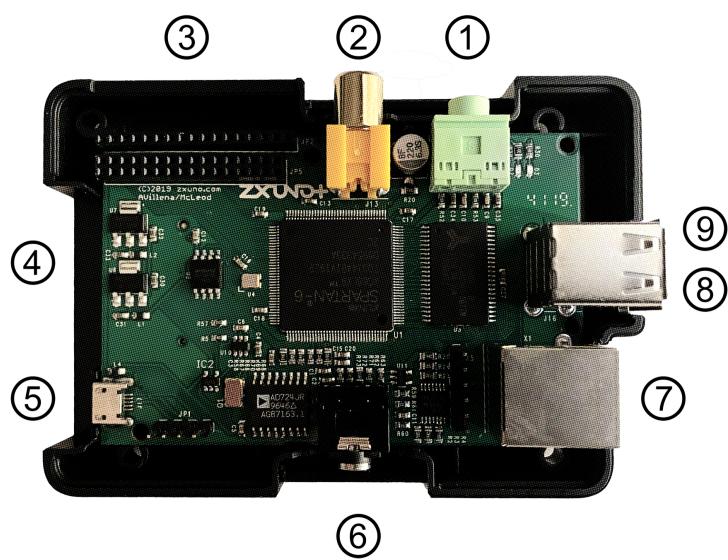
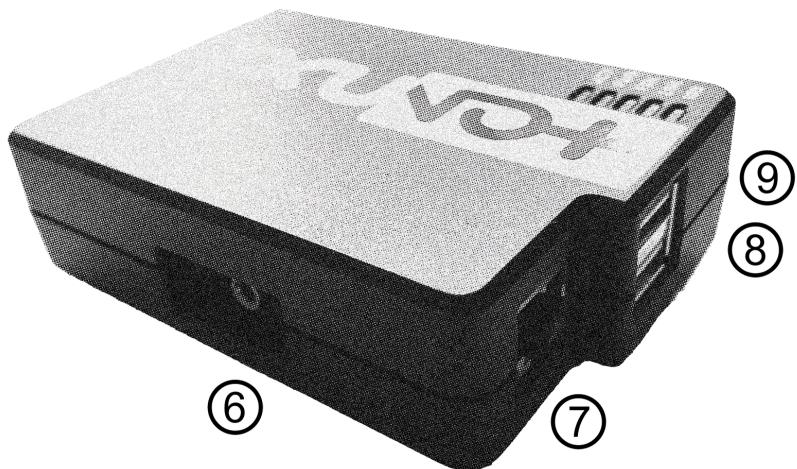
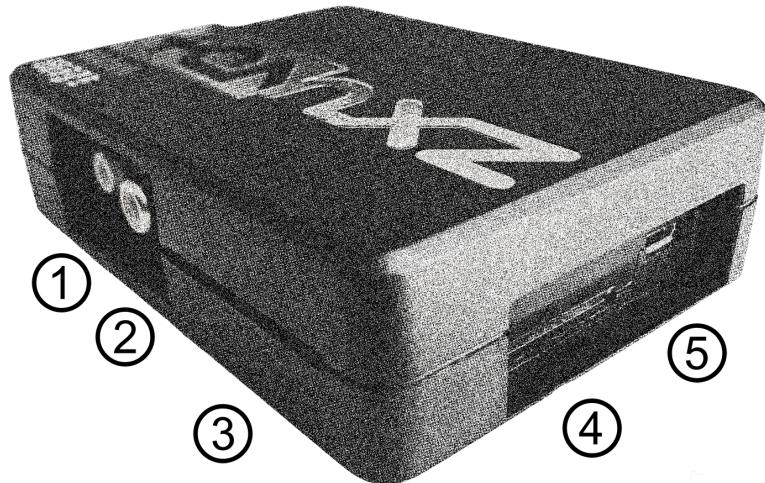
Most of the content is based on information previously shared:

- At [ZX-Uno forum](#)
- At [foroFPGA](#)
- Several existing FAQ, mostly the original version [by @uto\\_dev](#), and the latest one [by @desUBIKado](#)

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

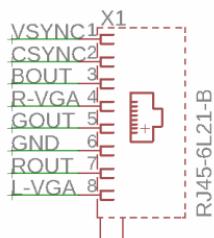


## Description

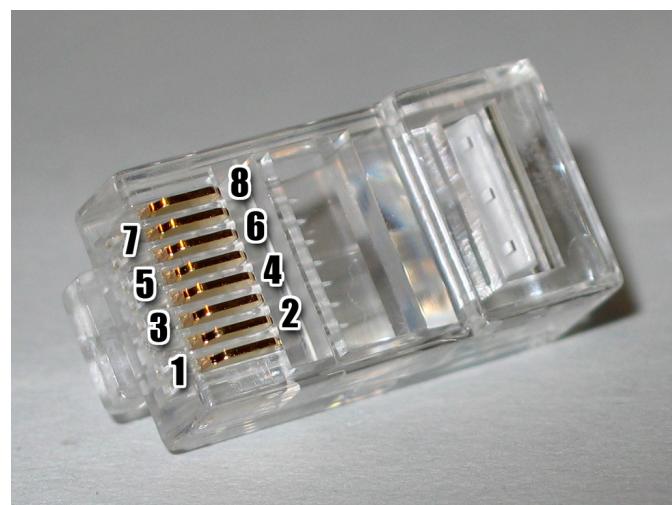
1	Audio Out
2	Video Out
3	Expansion Port
4	SD Card Slot
5	Power Socket
6	Audio In
7	Video Out (RJ-45)
8	Mouse USB (PS/2) Port
9	Keyboard USB (PS/2) Port

### RJ-45 Video Out

This is the pinout or RJ-45 connector to use as VGA output:



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



# 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**).

## Windows

For simple configurations, and cards of the correct 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.

## MacOS

For simple configurations, and cards of the correct 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
```

## 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.

## 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
loadpxz
playmid
playrmov
romsback
romsupgr
upgr16m
zxuc
zxunocfg
```

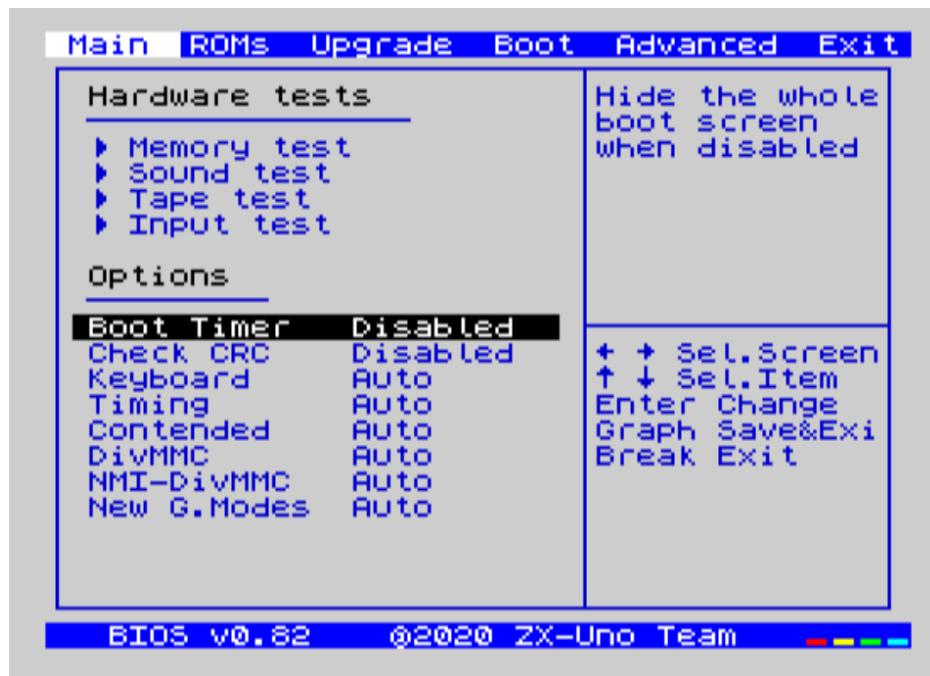
<<#\_zxuno+\_commands, It is explained later> what each of them does.

# 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.

## Main

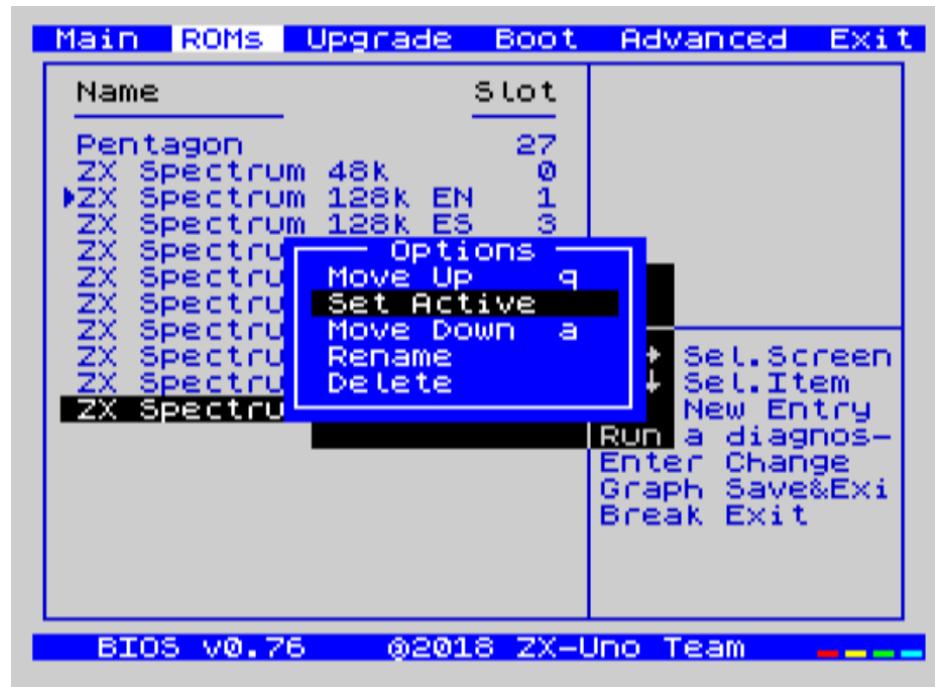


In the first configuration screen, in addition to being able to run several tests, you can define the default behavior for the following:

- 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)

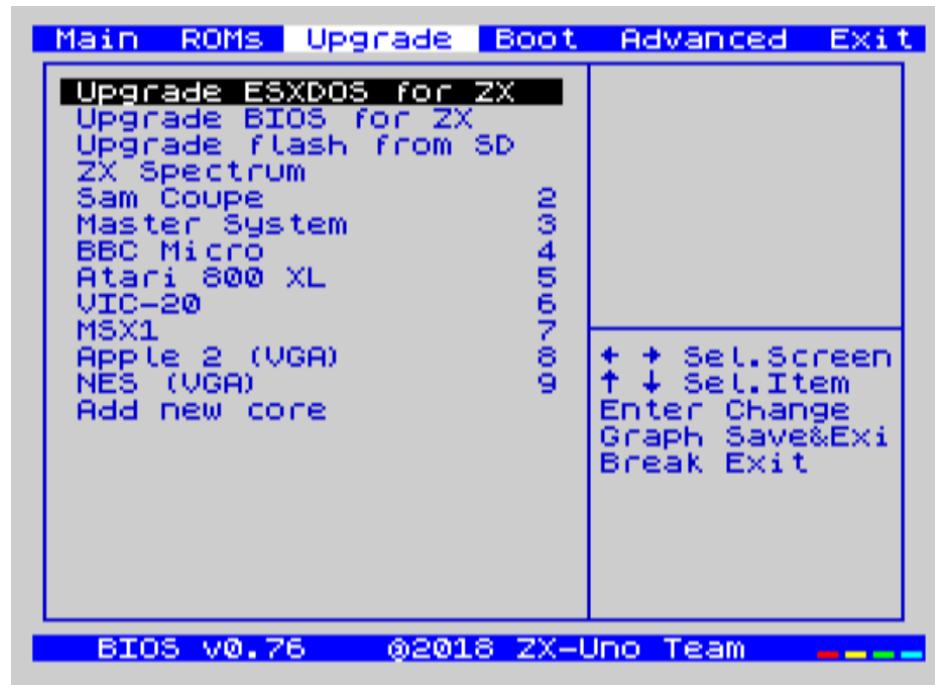
More technical information can be found on [de ZX-Uno Wiki](#).

ROMs



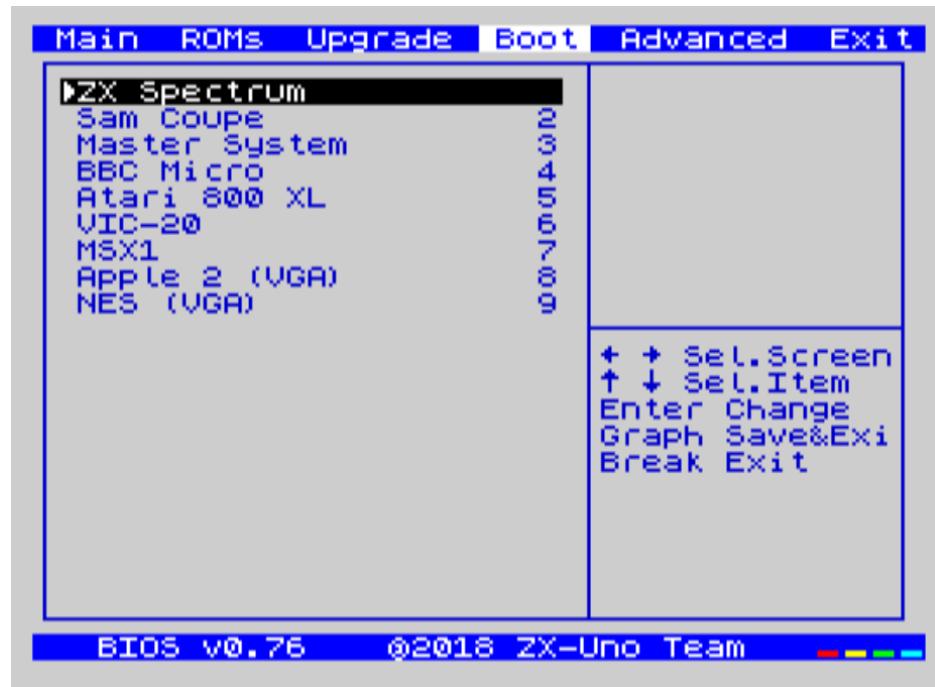
The second screen shows the installed ZX Spectrum ROMs. You can reorder (Move Up, Move Down), rename or delete each of them, as well as choose the one that will be loaded by default at startup (Set Active ).

## Upgrade



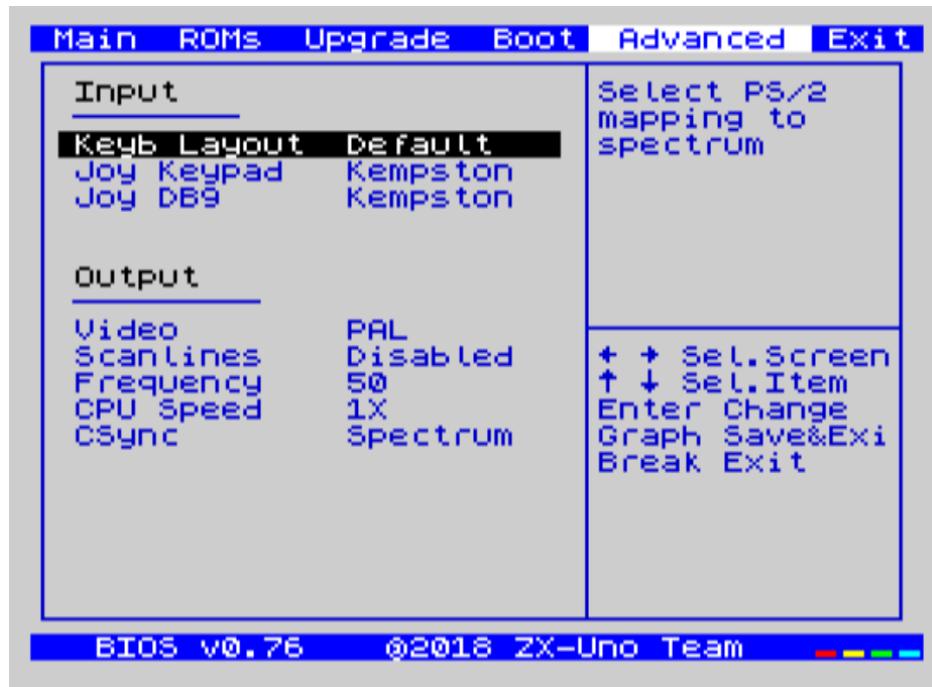
*Upgrade* screen is used to perform the different updates of the Flash memory content: esxdos, BIOS, Cores, etc. (see [the section corresponding to updates](#) for more information).

## Boot



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

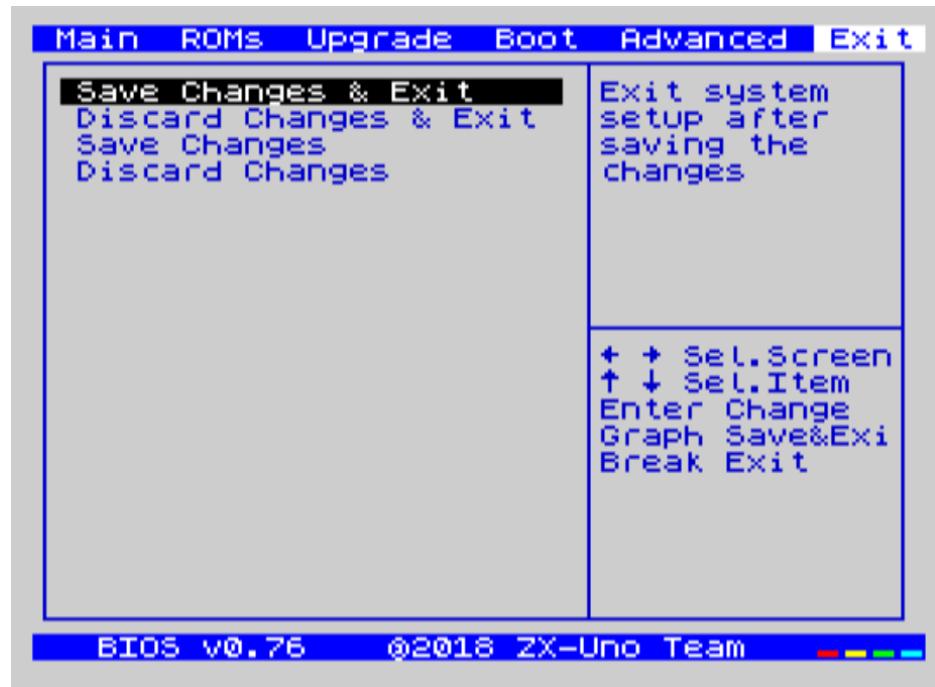
## Advanced



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): Kempston, Sinclair Joystick 1, Sinclair Joystick 2, Protek or Fuller
- Behavior of a joystick connected to the port (Joy DB9): Kempston, Sinclair Joystick 1, Sinclair Joystick 2, Protek, Fuller or simulate the keys Q, `A`, 0, `P`, Space and M
- 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

## 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 synchronisms 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
- 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 QAOOPSpC 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, 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.

# 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 the hard disk into several partitions to store data. ROM version 1.28 and later can share IDEDOS partitions with MBR partitions. In other cases, 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:          0xEF
 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](#).

With 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 needed 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 is 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 named "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](#).

# Keyboard

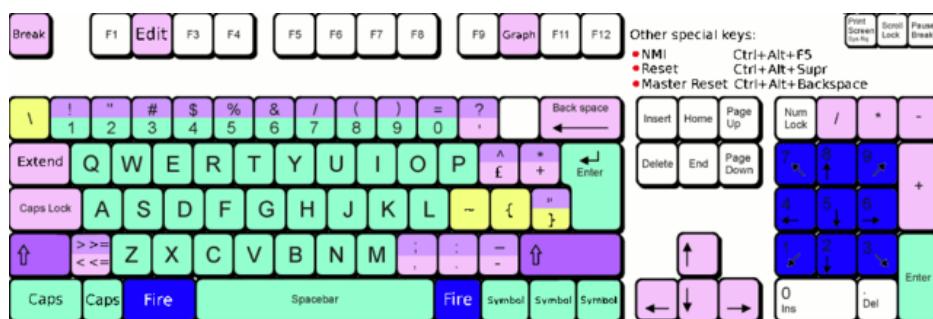
The keyboard map (physical keys of the keyboard assignment to the keystrokes that are presented to the different cores) is changed using the **Advanced** menu of the BIOS. There are three different maps to choose from: Spanish (default), English, and Spectrum (advanced).

You can also change it using the **keymap** utility. Inside **/bin** 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.

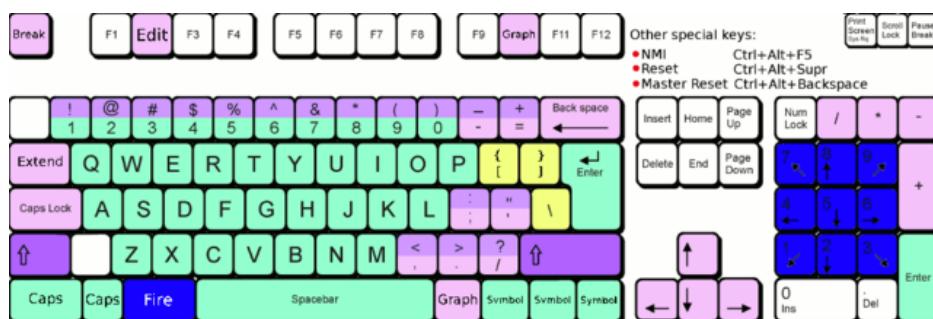
For the map to be preserved after a master reset, it has to be selected as **Default** in the BIOS.

For more information, see [this message in the ZX-Uno forum](#).

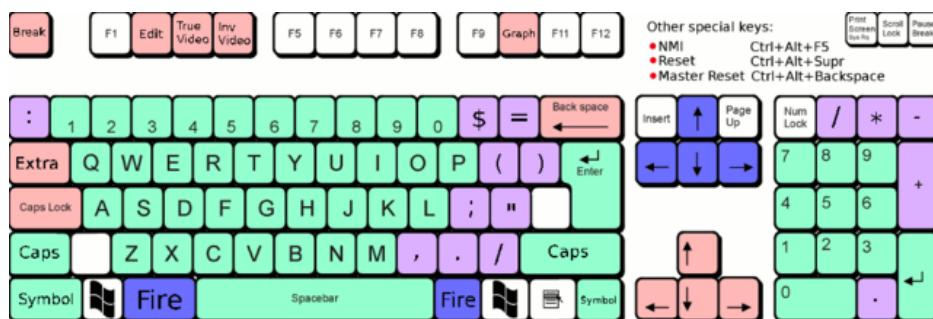
## Spanish



## English



## Spectrum



## Special keys and buttons

Special keys which can be used during startup:

- **F2**: 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.
- **/** (numeric keyboard): Load the default ZX Spectrum core ROM in "root" mode
- 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**: Edit
- **F5**: NMI
- **F7**: Play or pause when playing .PZX files
- **F8**: Rewind .PZX file to the previous mark
- **F10**: Graph
- **F12**: Turbo Boost. Speeds up CPU to 28MHz while pressed (beginning with core EXP27).
- **Ctrl+Alt+Backspace**: Hard reset. Backspace is the delete key, located in the top-right portion of the keyboard, above **Enter**.
- **Ctrl+Alt+Supr**: Soft reset.
- **Scroll Lock**: Switches between composite and VGA video modes.

# 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.

You can download the ROM, a user manual and other files from the [official Facebook Public Group](#).

Since it is 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 Z80 interpreter for Microsoft BASIC. SE Basic IV is designed to run on the [Chloe 280SE](#) but it can also run on ZX-Uno and similar computers.

SE Basic was originally conceived as a new firmware for the [ZX Spectrum SE](#). The earliest versions of SE Basic were patches applied to the original ZX Spectrum ROM, but later versions have been rewritten based on the [open source TS1000 / ZX81 improved ROM](#).

Version 3, also known as [OpenSE BASIC](#), is still maintained as an open source replacement firmware for the Spectrum. It's [included in the main Debian repository](#) for use with emulators.

Version IV is a fork of the previous version, done because there was no room left to add new features to the 16K ROM. The initial release (4.0 Anya) added another 16K ROM with support for Timex hi-res mode. The syntax was still largely Sinclair BASIC compatible at this point. Version 4.2 was rebuilt specifically for the Chloe 280SE, dropping support for legacy devices such as tape, adding full compatibility and integrated support for the esxdos kernel, and migrating to Microsoft BASIC syntax.

While it shares a common code base with many versions of Sinclair BASIC (the TS1000 ROM), it differs from them in significant ways:

- Code page (8-bit ASCII) support.
- Error message localization.
- 38 additional tokens.
- Standard terminal display (80 x 24).
- Terminal character entry (CTRL, META).
- Keyboard buffer.
- Full sized keyboard support.

The main differences from Microsoft BASIC are:

- Token abbreviation.
- On-entry syntax checking.
- Typically, brackets are optional.
- Motorola style number entry: % – binary @ – octal \$ – hexadecimal
- Always-on expression evaluation.
- Separate logical and bitwise operators.
- Automatic data typing.

You can find much more information, including the user manual, etc. at [the official wiki page](#).

## 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

# esxdos

## 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**: Para 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 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
- **snapshot**: 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 load 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.

## 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 utilility for certain features of ZX-Uno such as timings, contention, keyboard type, CPU speed, video type or vertical frequency

# Making RDM (RaDastan Movie) files

The `PLAYMOV` "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 (`makevideoradas.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 `makevideoradas`, you need another two tools: `ffmpeg` and `imagemagick`. These can be installed with a package manager (`apt`, `yum`, `pacman`, `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](#).

# 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.



At this moment, **romsupgr**, only works correctly with RomPack files using a maximum of 35 slots.



Remember that if the ZXUNO+ is started by 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.



At this moment, **romsback**, only stores correctly the first 35 used slots.

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 `ESXDOS...` 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 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

### 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).

### Keyboard

#### Special keys and buttons

While the core is running:

- **Esc**: BREAK
- **F5**: NMI
- **F8**: 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 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).

## Keyboard

### Special keys and buttons

While the core is running:

- **Esc**: BREAK
- **F5**: NMI
- **F8**: 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.

# 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:** Soft Reset
- **F1:** Turbo mode 1Mhz
- **F2:** Turbo mode 2Mhz
- **F3:** Turbo mode 4Mhz
- **F4:** Turbo mode 8Mhz

The keyboard uses the following mapping:



# Acorn Electron

The [Acorn Electron](#) was a budget version of the BBC Micro educational/home computer. The core is based on el [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

And 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 some any program that can defrag files or FAT filesystems 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.

## 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**: 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
```

## Basic Guide

Sometimes, after starting up the core, a screen full of @ appears. Ejecting and inserting, or only inserting, the SD card will fully start the system.



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

# 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

## 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**: Reset.
- **Ctrl+Alt+Backspace**: Hard reset. Backspace is the delete key, located in the top-right portion of the keyboard, above **Enter**.
- **End**: Switches between 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 successor 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, 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

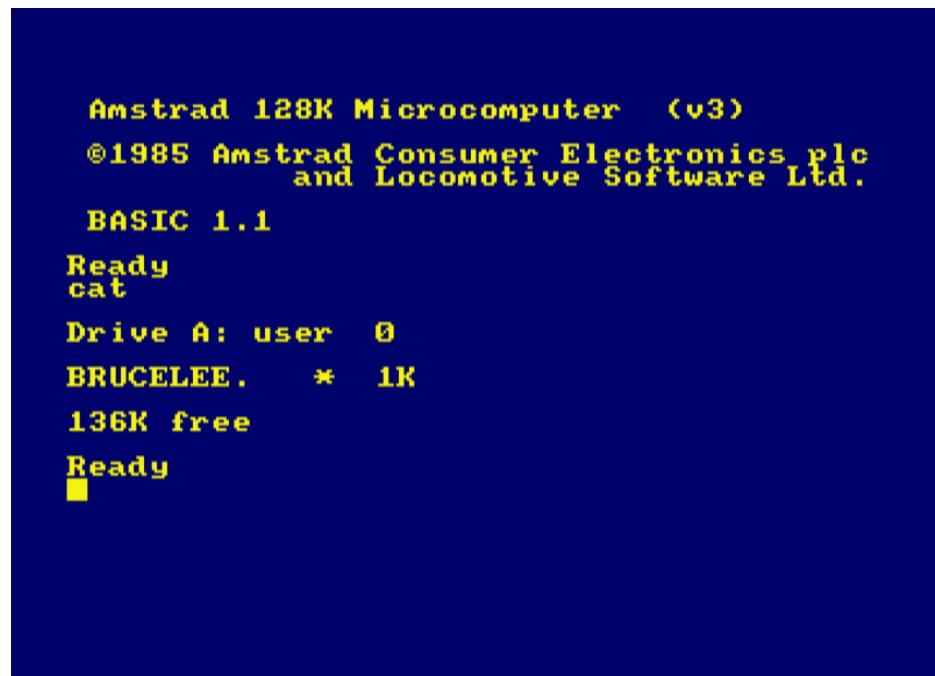
### Special keys and buttons

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.



```
Amstrad 128K Microcomputer (v3)
©1985 Amstrad Consumer Electronics plc
and Locomotive Software Ltd.

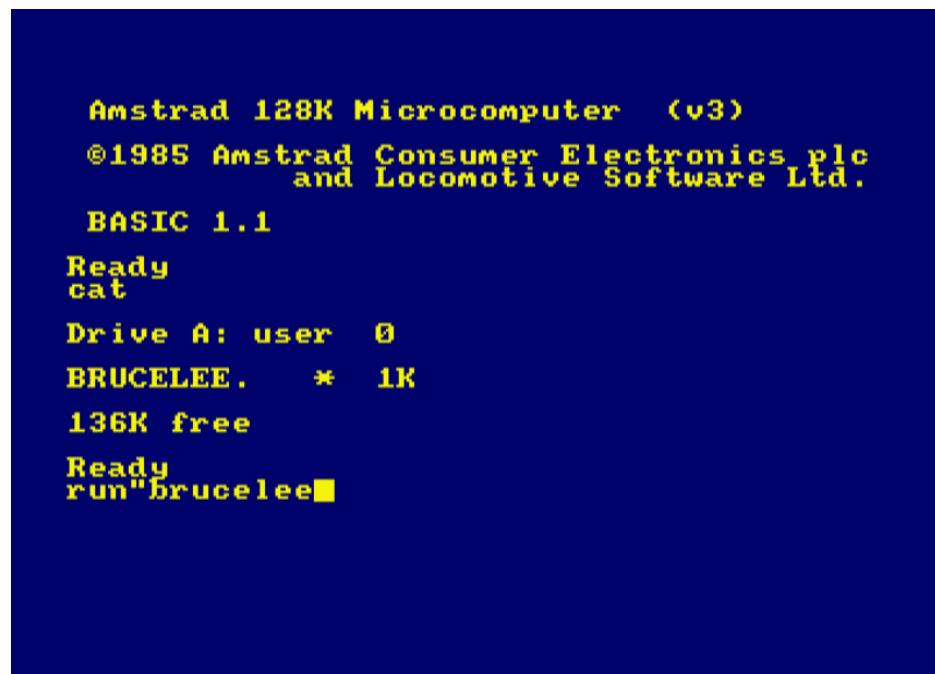
BASIC 1.1

Ready
cat

Drive A: user 0
BRUCELEE.* 1K
136K free

Ready■
```

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



```
Amstrad 128K Microcomputer (v3)
©1985 Amstrad Consumer Electronics plc
and Locomotive Software Ltd.

BASIC 1.1

Ready
cat

Drive A: user 0
BRUCELEE.* 1K
136K free

Ready
run"brucelee■
```

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

# 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 buttons)
- 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 **D0**), 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

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**: Hard reset. Backspace is the delete key, located in the top-right portion of the keyboard, above **Enter**
- **F12**: 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 90°)
- Inverted Verticale (the machine used a monitor rotated 270°)
- 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 90° rotation of directional controls
- **Scroll Lock**: Switches between composite and VGA video modes.
- **F10**: Soft Reset
- **Ctrl+Alt+Backspace**: 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

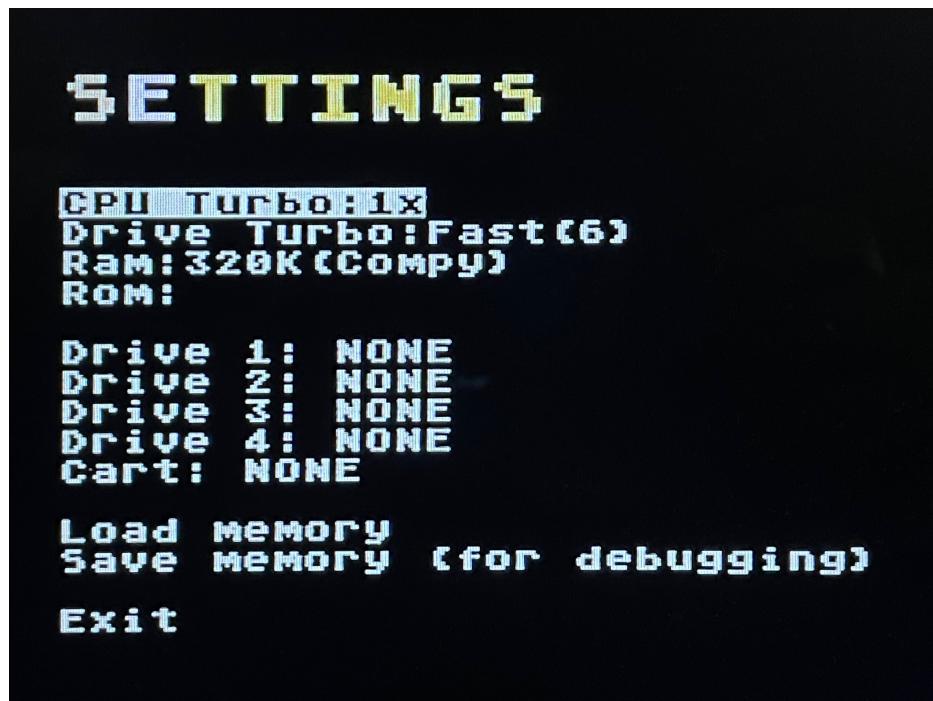
### Special keys and buttons

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`: Hard reset.
- `F5`: Help
- `F6`: Start
- `F7`: Select
- `F9`: Reset
- `F11`: Load a disk
- `F12`: Show or hide the configuration menu
- The numeric keypad emulates a joystick, where `5` and `2` keys work as *down* direction and `0` is the fire button

## Basic Guide

Pressing **F12** 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**: Hard reset.

## 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

# 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.

# 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 Commodore 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 Kernel 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, using 256K blocks. Download 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 imagen1.d64 dummyto256.bin imagen2.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**: Change joystick between port 0 and port 1
- **F12**: Soft Reset
- **Ctrl+Alt+Backspace**: Hard reset.

## Basic Guide

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

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 **Shift+TAB** or use the command **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 / RGB-Compuesto
- 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 imagen1.d64 + dummyto256.bin + imagen2.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 imagen1.d64 dummyto256.bin imagen2.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

Durante la ejecución del core:

While the core is running:

- **Esc**: Start/Stop
- **Tab**: RUN/STOP
- **Alt** = Commodore
- Numeric Keyboard: Emulated Joystick
- **F1** to **F8**: F1 to F8
- **F9**: Pound symbol key
- **F10**: + 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**: 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**: Hard reset.

## Basic Guide

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

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**.

# 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=Frances, 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**: Enables and disables turbo mode
- **Ctrl+Alt+Supr**: Soft Reset
- **Ctrl+Alt+F12**: Hard Reset
- **Ctrl+Alt+Backspace**: Restarts the FPGA
- **Left ALT**: MSX GRAPH
- **Right ALT**: MSX CODE
- **Page Up**: MSX SELECT
- **Start**: 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.

# 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

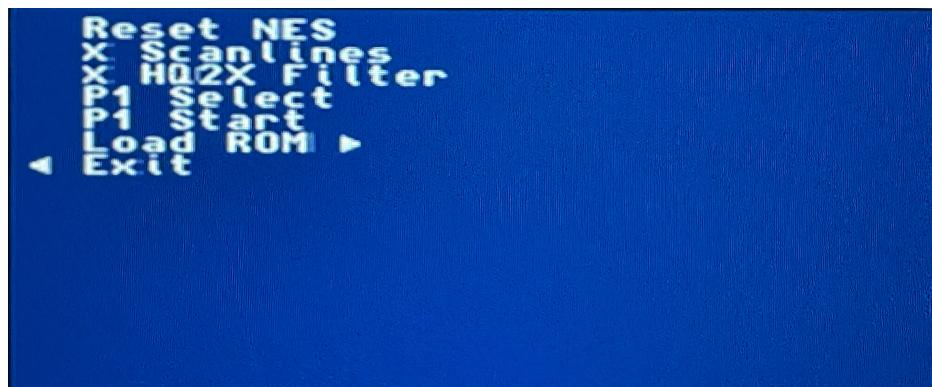
### Special keys and buttons

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**: Hard reset

## 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

# PC XT

[Next186lite](#) core is an implementation of [Next186](#) core for FleaFPGA and Papilio Pro, pero 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 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](#).

See the [corresponding section](#) for instructions of how to install the PC XT core in ZXUNO+.

# SmartROM

SmartROM is a kind of firmware that allows to 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 [instalado esxdos 0.8.8](#) into the card, and also 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+.

# 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.

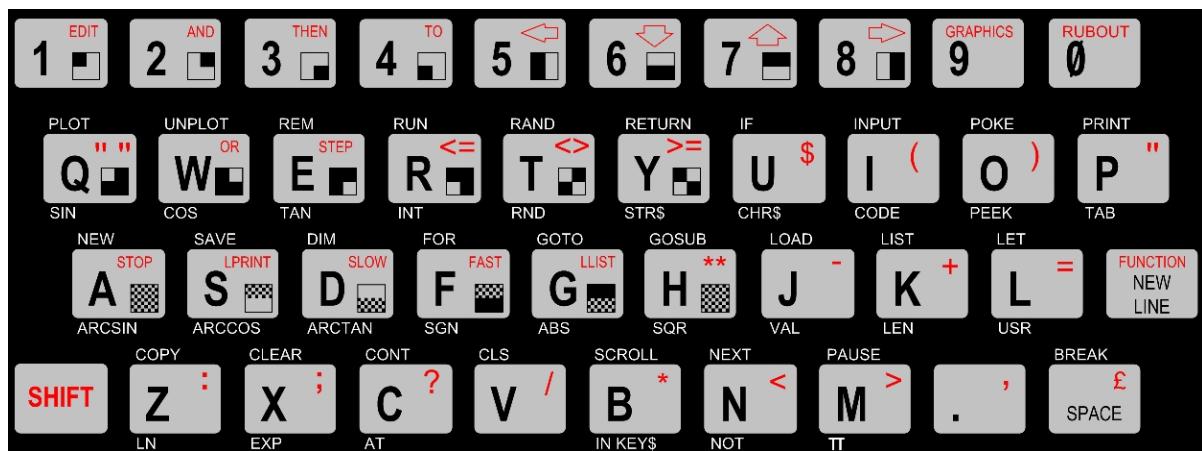
The ZXUNO+ version has been made by jepalza based on a previous ZX97 implementation.

## 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 an external tape (or from another external audio device) with the command LOAD"". Take note, that, while loading, the video output is disabled and, unlike the original machine, you can hear the loading sound.



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

# Other Hardware

## 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 stero 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 examle, 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 examle, 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 `txzplay` 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.

[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 ohmis. 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`.

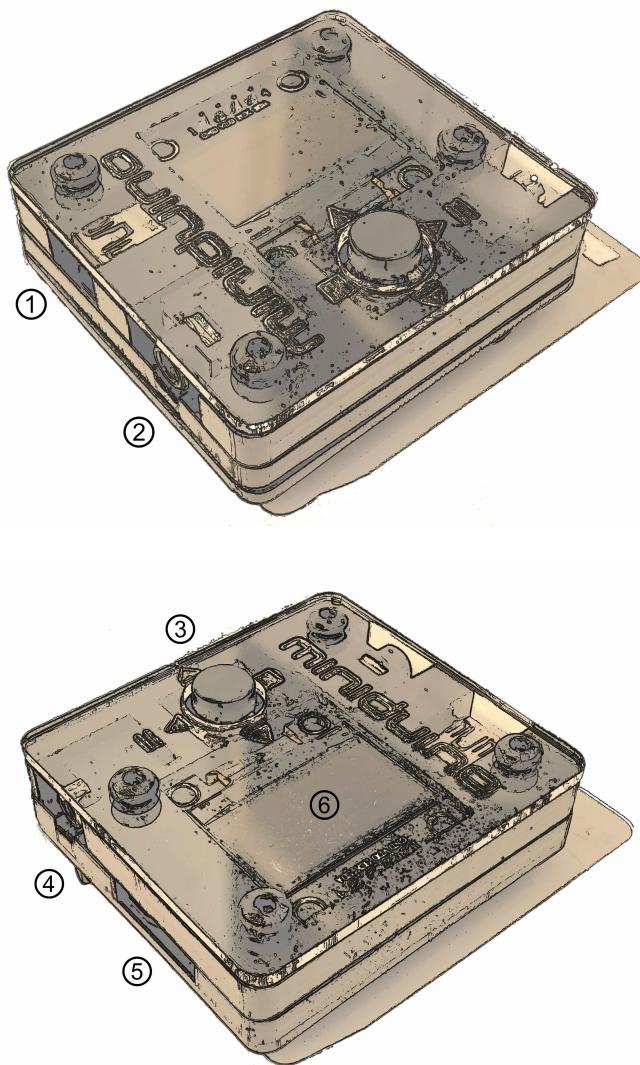
`tsx2wav` in [TSXphpclass](#) is made with PHP and that can export from `TSX` to `WAV`.

## Miniduino

[Miniduino](#) is a tape file audio player, based on a STM32F103C8T6 microcontroller with 64KB flash memory, and [Maxduino](#) firmware preinstalled.

Maxduino plays, in a very similar way to how [cassette tape](#) players worked, digital tape files in format as like **TAP** and **TZX** (ZX Spectrum), **O** (ZX80), **P** (ZX81), **CDT** (Amstrad CPC), **CAS**(MSX) **TSX** (MSX, Acorn, etc). It is also possible to play AY music files as if they were tapes, in order 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

A SD is needed in order 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 too.

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

Besides the card, you have to plug an appropriate audio cable to [ZXUNO+ audio input](#). It must have a 3.5 mm stero 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 2.6 mm jack.

Copy the tape files ([TAP](#), [TZX](#), [O](#), [P](#), [CAS](#), [TSX](#), etc) to the first partition of the SD card. They can be organized using folders or directories.



The player 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.

## Use

Once the SD card with the data files is inserted, it's turned on plugging in the included USB power cable.



Pressing down the control button shows the options menu which can set the following:

- 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, etc.)
- Converter (TSXCzxpUEFWS): Enables turbo loading **.CAS** and **.TSX** files, changes signal for Spectrum and Amstrad CPC files and/or change parity when playing Acorn Electron and BBC Micro **.UEF** files
- (Skip BLK): To disable (Skip ON) or enable automatic pause when 2A blocks are found

When not inside the options menu, the control button is used as a four directional control joystick, which has two different behaviours depending 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 directories list
- Left (Stop) goes one level up in the directory tree
- Right (Play/Pause) enters into a directory or, if the selection is a file, tries to play it

Once a file is being played, stop playing it with the left button (Stop) or pause using the right button (Play/Pause).



When in pause (tape block browser):

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

## Maxduino firmware upgrade

Maxduino firmware is periodically updated and improved. You can track the changes and improvements either at the [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.

### Environment setup

Firmware flashing is done from a computer (Windows, Mac, Linux) with [Arduino IDE](#) installed.

You have to 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:

First, adding ARM Cortex M3 support from menu Tools → board → board manager, and installing "Arduino SAM boards (Cortex-M3)"

Then, you have to add STM32 microcontroller support, downloading the file available at [this link](#).

Extract the contents to the current user directory in

```
...Arduino/hardware/Arduino_STM32
```

If 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 does not appear as USB device in Arduino ID when plugged, it may be necessary to install [libusb](#).

Finally, when on Mac or Linux, the file `maple_upload` inside `Arduino_STM32` has to be changed with a text editor. Those lines do not work:

```
if [ $# -eq 5 ]; then
    dfuse_addr="--dfuse-address $5"
else
    dfuse_addr=""
fi
```

And have to be changed into this:

```
dfuse_addr=""
```

## Upgrade

Once you have the environment ready, download the software from the [official repository in GitHub](#)



Minduino player with STM32 microcontroller is only supported from 1.65 and up

Load the project file with Arduino IDE (for example `MaxDuino_v1.66.ino`).

Check in the file `userSTM32Config.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 computer using the USB cable, and find the assigned port, normally with a name like "Maple Mini" (for example: COM5 Maple Mini)

Set the following options in menu "":

```
Board: Generic STM32F104C Series
Variant: STM32F104C8 (20k RAM, 64k Flash)
Upload Method: STM32duino bootloader
CPU Speed: 72Mhz (Normal)
Optimize: Smallest (default)
Port: <Previously identified port>
```

Finally, 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.

# 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, ZX DOS 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 exteacted 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,xdlh17,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:

<code>i</code>	Keyboard issue 3 enabled (instead of issue 2)
<code>c</code>	Disable memory contention
<code>d</code>	Enable DivMMC
<code>n</code>	Enable NMI DivMMC (esxdos Menu)
<code>p</code>	Use Pentagon Timings
<code>t</code>	Use 128K timings
<code>s</code>	Disable DivMMC and ZXMMC ports
<code>m</code>	Enable Timex Horizontal MMU
<code>h</code>	Disable ROM high bit (1FFD bit 2)
<code>l</code>	Disable ROM low bit (7FFD bit 4)
<code>1</code>	Disable 1FFD port (+2A/3 paging)
<code>7</code>	Disable 7FFD port (128K paging)
<code>2</code>	Disable TurboSound (secondary AY chip)
<code>a</code>	Disable AY chip
<code>r</code>	Disable Radastanian mode
<code>x</code>	Disable Timex mode
<code>u</code>	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 ussing 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
```

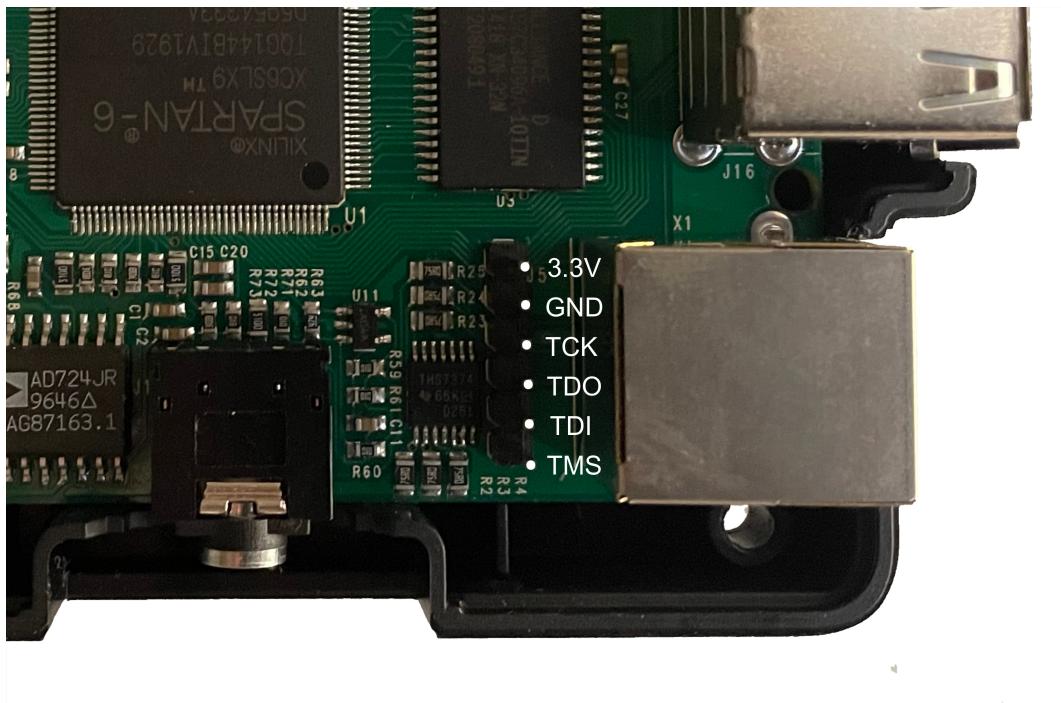
# Firmware recovery

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**

## 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](#)

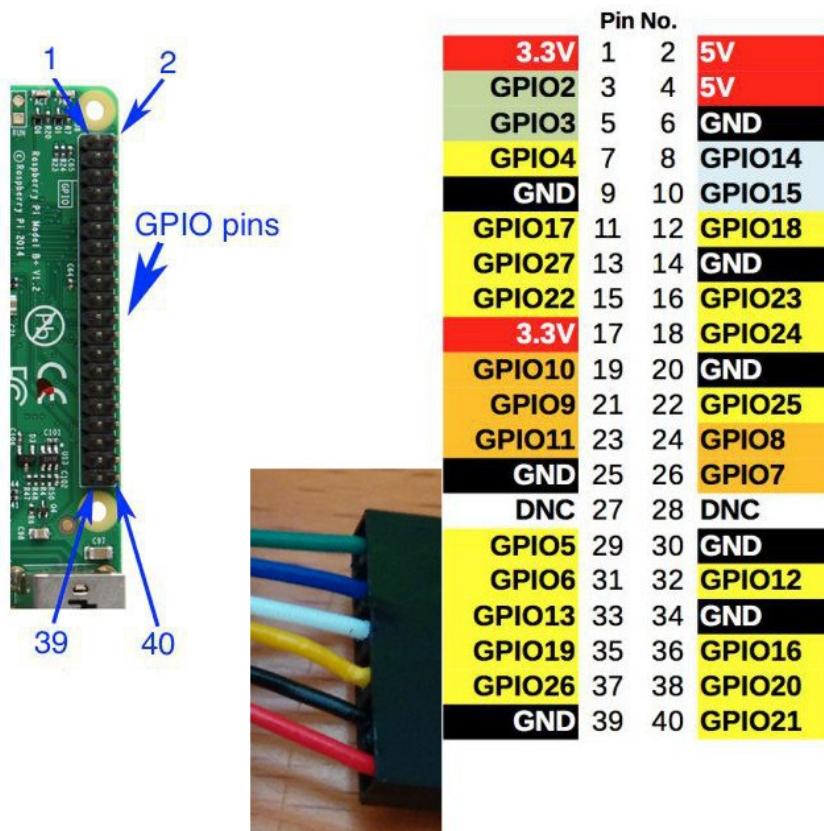
### Instruction Steps:

1. Install Raspberry Pi OS (formely 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.

If 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:

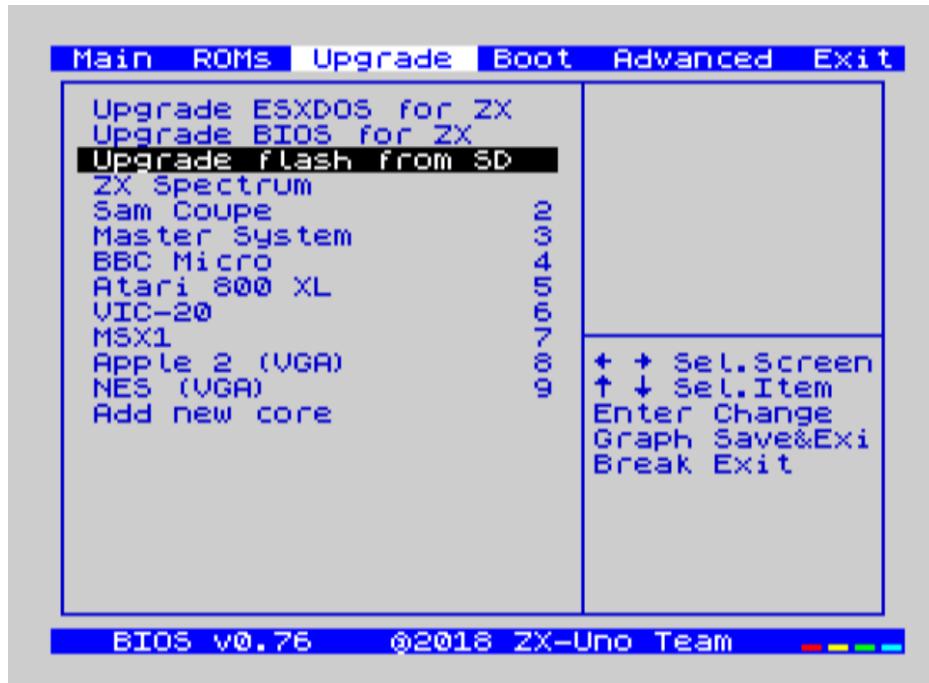
```
cd /home/pi/zxunoplus/unbrick  
sudo openocd -f /home/pi/zxunoplus/unbrick/raspberrypi2-native.cfg -f  
/usr/local/share/openocd/scripts/cpld/xilinx-xc6s.cfg -c "init; xc6s_program xc6s.tap;  
pld load 0 recovery.bit ; exit"
```

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"
```

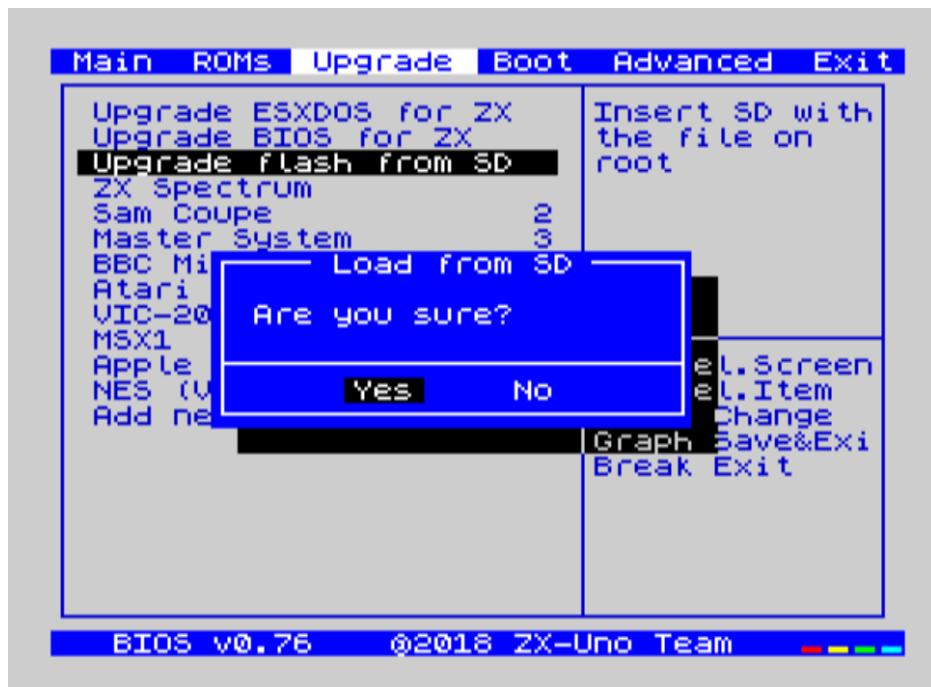
11. If all goes well, we will see that the FPGA LED change their state and the BIOS is shown on the display.

If there is no image on the display, press **Scroll Lock** 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.

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** 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.

# References

[ZX-Uno](#)

[ZX-Uno FAQ](#)

[Guía rápida del ZX-Uno](#)

[Core ZX Spectrum](#)

[The ZX Spectrum +3e Homepage](#)

[Sharing a +3e disk with PC \(FAT\) partitions](#)

[Layouts de teclado](#)

[Firmware de teclado para ZX Go+](#)

[Almost \(In-\) Complete List of esxDOS DOT-Commands](#)

[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](#)

[Odtyssey Speech And Sound Effects Module Manual](#)

[Programming a Spartan 6 with a Raspberry Pi](#)

[Tutorial para desbriquiar el ZX-Uno con una Raspberry](#)

[Como programar un UnAmiga con la Raspberry Pi \(o Linux\) con el USB-Blaster y OpenOCD](#)