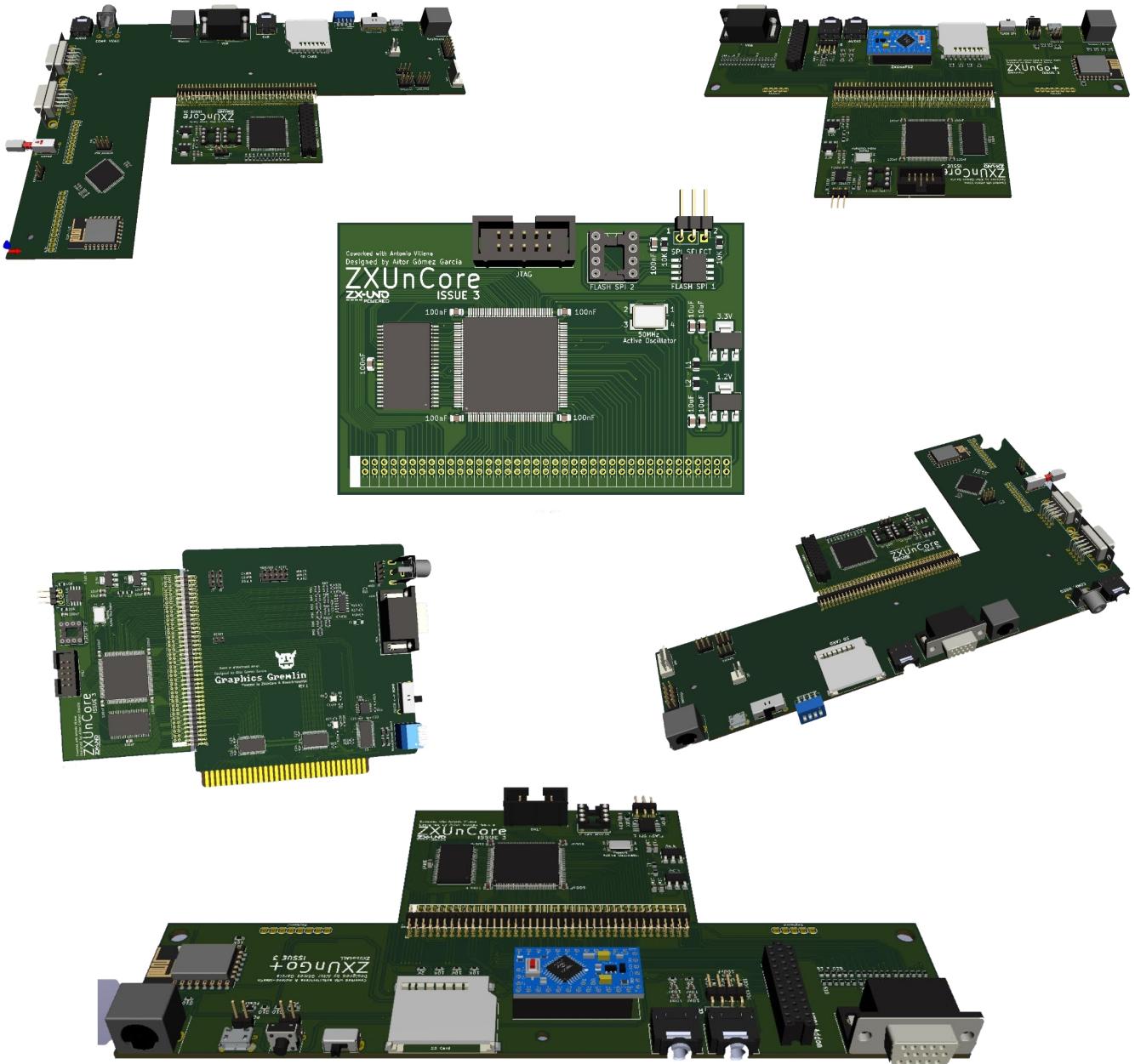


ZXUno4ALL

Manual



Index

Introduction	1
Aknowledgements	1
Modules	2
ZXUnCore	2
ZXUnGo+ DB9	3
ZXUnGo+	5
Ports and Connectors	6
Description	6
Initial Setup	7
SD card formatting	8
Windows	10
macOS	11
Linux	13
Keyboard	14
Membrane keyboard	14
Keyboard modes	15
Functions	17
Firmware update	18
PS/2 keyboard	21
Special keys and buttons	22
esxdos	24
BIOS	26
Main	27
ROMs	28
Upgrade	28
Boot	29
Advanced	30
Exit	32
ZX Spectrum	33
ROMs	35
DerbyPro	35
CargandoLeches	37
POKEs	37
Preparing ultrafast loading tapes	38
SE Basic IV	39
Other ROMs	41
SD advanced format (+3e)	42
Windows	42

macOS	42
Linux	44
+3e	47
esxdos Commands	48
Basic Guide	48
ZX-Uno commands	50
Wi-Fi	51
Network tools for ZX-Uno pack	51
FTP-Uno	51
UART Terminal	52
Making RDM (RaDastan Movie) files	53
FUZIX	54
How to Build	54
How to use	55
Upgrade	59
BIOS	59
ROMs	59
Cores	59
esxdos	60
Flash Memory	61
Other cores	62
ZX Spectrum EXP27 without PZX	62
ZX Spectrum 48K (Kyp)	62
SD card format	62
Keyboard	62
Special keys and buttons	62
ZX Spectrum 128K (Kyp)	63
SD card format	63
Keyboard	63
Special keys and buttons	63
ZX Spectrum (Kyp/azesmbog)	64
SD card format	64
Keyboard	64
Special keys and buttons	64
ZX Spectrum TBBBlue	65
SD card format	65
Keyboard	66
Special keys and buttons	66
Basic Guide	66
Acorn Atom	68
SD card format	68

Keyboard	69
Special keys and buttons	69
Basic Guide	70
Acorn Electron	71
SD card format	71
Keyboard	72
Special keys and buttons	72
Basic Guide	73
Amstrad CPC 464	75
SD card format	75
Keyboard	75
Special keys and buttons	75
Basic Guide	75
Amstrad CPC 6128	77
SD card format	77
Keyboard	77
Special keys and buttons	77
Basic Guide	78
Apple I	79
SD card format	79
Keyboard	79
Basic Guide	80
Apple II	82
SD card formatting	82
Windows	82
macOS and Linux	82
Keyboard	83
Special keys and buttons	83
Arcades	84
Keyboard	84
Special keys and buttons	84
Atari 800XL	85
SD card	85
Keyboard	85
Special keys and buttons	85
Basic Guide	86
Atari 2600	87
SD card format	87
Keyboard	87
Special keys and buttons	87
Control customization	88

Basic Guide	88
BBC Micro	89
SD card format	89
Keyboard	90
Special keys and buttons	90
Basic guide	91
Computers Lynx	93
SD card format	93
Keyboard	93
Special keys and buttons	93
Basic Guide	94
CHIP-8	95
SD card format	95
Keyboard	95
Special keys and buttons	95
Special keys and buttons	95
Basic Guide	96
Colecovision	97
SD card format	97
Keyboard	97
Special keys and buttons	97
Basic Guide	98
Colour Genie	99
SD card format	99
Keyboard	99
Special keys and buttons	99
Basic Guide	100
Commodore 16	102
SD card format	103
Windows	103
macOS and Linux	103
Keyboard	104
Special keys and buttons	104
Basic Guide	104
Commodore 64	106
SD card format	107
Windows	107
macOS and Linux	107
Keyboard	108
Special keys and buttons	108
Basic Guide	109

Commodore PET	110
SD card format	110
Keyboard	110
Special keys and buttons	110
Basic Guide	111
Commodore VIC-20	112
SD card format	112
Keyboard	112
Special keys and buttons	112
Basic Guide	112
Flappy Bird	114
SD card format	114
Keyboard	114
Special keys and buttons	114
Galaksija	115
SD card format	115
Keyboard	116
Special keys and buttons	116
Basic guide	117
HT-1080Z	118
SD card format	118
Keyboard	119
Special keys and buttons	119
Basic Guide	120
Jupiter ACE	122
SD card format	122
Keyboard	122
Special keys and buttons	122
Basic Guide	123
MSX	124
SD card format	124
Keyboard	125
Special keys and buttons	125
Basic Guide	126
MSXCTRL	127
Other	127
NES	128
SD card format	128
Keyboard	128
Special keys and buttons	128
Control customization	129

Basic Guide	130
Ondra SPO 186	131
SD card format	131
Keyboard	131
Special keys and buttons	131
Basic guide	132
Oric Atmos	133
SD card format	133
Windows	133
macOS and Linux	133
Keyboard	134
Special keys and buttons	134
Basic Guide	135
Oric Atmos (Kyp)	136
SD card format	136
Keyboard	137
Special keys and buttons	137
Basic Guide	138
PC XT	140
SD card format	140
Windows	141
Linux	141
macOS	141
Keyboard	142
Special keys and buttons	142
Basic Guide	143
PC XT CGA	144
SD card format	144
Keyboard	145
Special keys and buttons	145
Basic Guide	146
Pong	147
SD format	147
Keyboard	147
Special keys and buttons	147
Basic Guide	148
SAM Coupé	149
SD card format	149
Keyboard	149
Special keys and buttons	149
Basic Guide	150

Sega Master System	151
SD card format	151
Keyboard	151
Special keys and buttons	151
Basic Guide	152
SmartROM	153
SD card format	153
Vectrex	154
SD card format	154
Keyboard	154
Special keys and buttons	154
Basic guide	155
Videopac	156
SD card format	156
Keyboard	156
Special keys and buttons	156
Basic Guide	157
Change VDC ROM charset	158
ZX81	159
SD card format	159
Keyboard	159
Basic Guide	159
ZX81 and ZX80	160
SD card format	160
Keyboard	161
Special keys and buttons	162
Basic Guide	162
Other Hardware	164
Joy2PS2	164
Firmware upgrade	165
Loading from tape	166
Cassette Player	166
Computer	166
PlayTZX	166
Mobile phone, tablet, MP3 player, etc.	168
Audio file conversion	168
Troubleshooting	169
Firmware images management	169
zx123_tool	169
Firmware recovery	173
Recovery using audio	173

JTAG cable connections	174
Recovery using a Raspberry Pi	175
Recovery using macOS and USB-Blaster cable	181
Recovery using Windows and USB-Blaster cable	186
References	190
Scan Codes	190
References	191
Spectrum	191
Scan Codes	191
ZX-Uno control I/O registers	192
\$00 MASTERCONF	193
\$01 MASTERMAPPER	194
\$02 FLASHSPI	194
\$03 FLASHCS	194
\$04 SCancode	194
\$05 KEYSTAT	195
\$06 JOYCONF	195
\$07 KEYMAP	195
\$09 MOUSEDATA	196
\$0A MOUSESTATUS	196
\$0B SCANDBLCTRL	197
\$0C RASTERLINE	198
\$0D RASTERCTRL	198
\$0E DEVCONTROL	199
\$0F DEVCTRL2	200
\$10 MEMREPORT	200
\$40 RADASCTRL	201
\$41 RADASOFFSET	201
\$42 RADASPADDING	201
\$43 RADASPALBANK	202
\$80 HOFFS48K	202
\$81 VOFFS48K	202
\$82 HOFFS128K	202
\$83 VOFFS128K	203
\$84	203
\$85 VOFFSPEN	203
\$A0 DMACTRL	203
\$A1 DMASRC	203
\$A2 DMADST	203
\$A3 DMAPRE	204
\$A4 DMALEN	204

\$A5 DMAPROB	204
\$A6 DMASTAT	204
\$C6 UARTDATA	204
\$C7 UARTSTAT	204
\$C8 - \$DF RESERVED	204
\$F0 SRAMADDR	205
\$F1 MADDRINC	205
\$F3 VDECKCTRL	205
\$F7 AUDIOMIX	205
\$FB AD724	206
\$FC COREADDR	206
\$FD COREBOOT	206
\$FE SCRATCH	206
Links	208

Introduction

ZXUno4All is a variant, made by spark2k06, de [ZX-Uno](#) a hardware and software project based on an FPGA board programmed to work like a ZX Spectrum computer, and created by the ZX-Uno team: Superfo, AVillena, McLeod, Quest and Hark0.

Over time, the project has been growing, and now it is possible to install different configurations (cores) in the flash memory of the FPGA, which work like different systems than the ZX Spectrum, and you can choose to start the ZXUno4All with the desired configuration among all those installed.

ZX-Uno official web page is <https://zxuno.speccy.org>.

ZXUno4All concept is to give to everyone all the cores and software for ZX-Uno, using backplanes in different formats, which use the main module [ZXUnCore](#). This one is made in such a way that derived works with no direct relation to ZX-Uno. This manual is mainly oriented to the use of [ZXUnGo+](#) or [ZXUnSugarless](#), along with [ZXUnCore](#).

In general, this document will talk about ZXUnGo+, explaining the differences with other when necessary.

ZXUno4ALL files are available at the <https://github.com/spark2k06/ZXUno4ALL> repository.

You can also find information about the evolution of the project at Va de Retro forums:

<https://www.va-de-retro.com/foros/viewtopic.php?f=63&t=9201>

<https://www.va-de-retro.com/foros/viewtopic.php?f=63&t=9248>

Acknowledgements

Most of the content is based on information previously shared:

- At [ZX-Uno forum](#)
- At [foroFPGA](#)
- Several existing FAQ, mostly the original version [by @uto_dev](#), and the latest one [by @desUBIKado](#)
- At [ZX-Uno official Telegram channel](#)

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

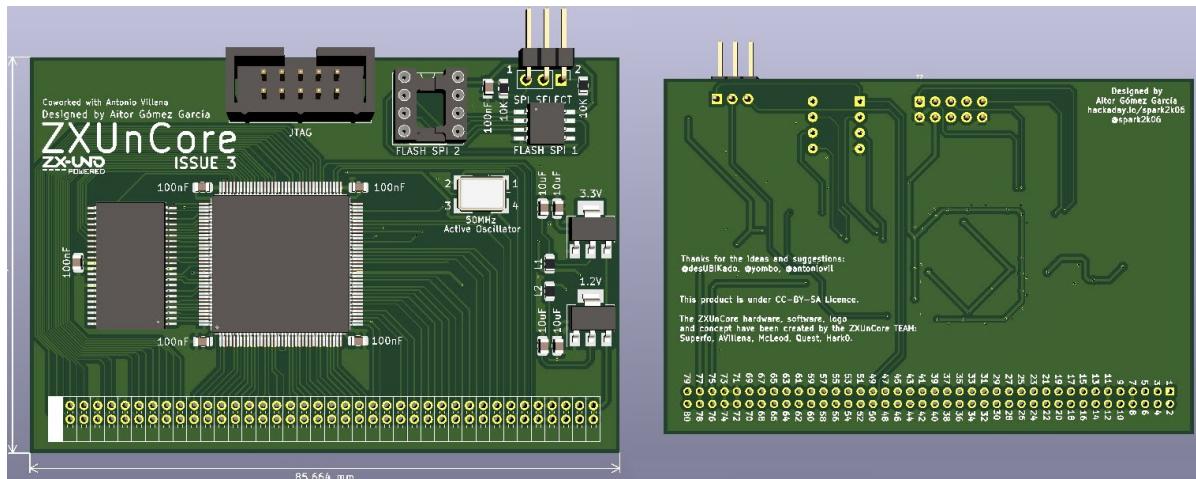
Without the previous work of all of these people (and more), this manual wouldn't exist.

Modules

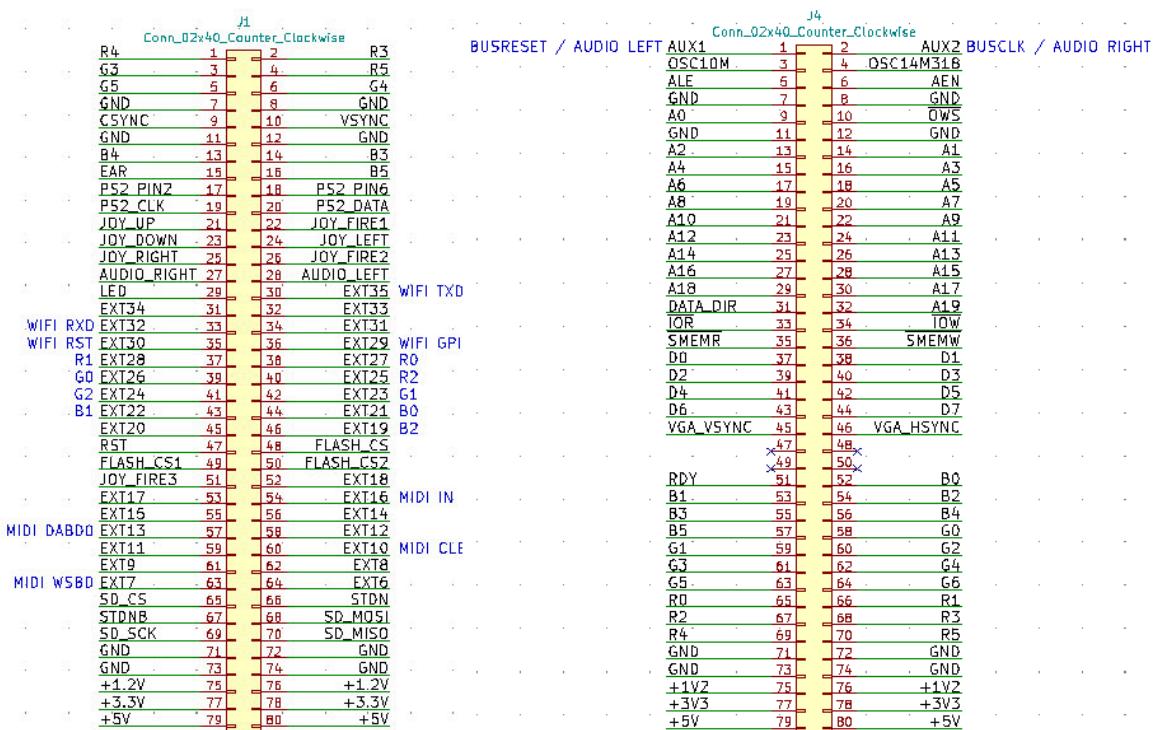
ZXUnCore

Main module, and the base for other modules. It's basically a stripped ZX-Uno with the basic hardware and as a module. Its main features are:

- RGB666 Video DAC support
 - Integrated SRAM memory (512Kb or 2Mb).
 - Hot selectable Double Flash SPI, and one of them as a slot

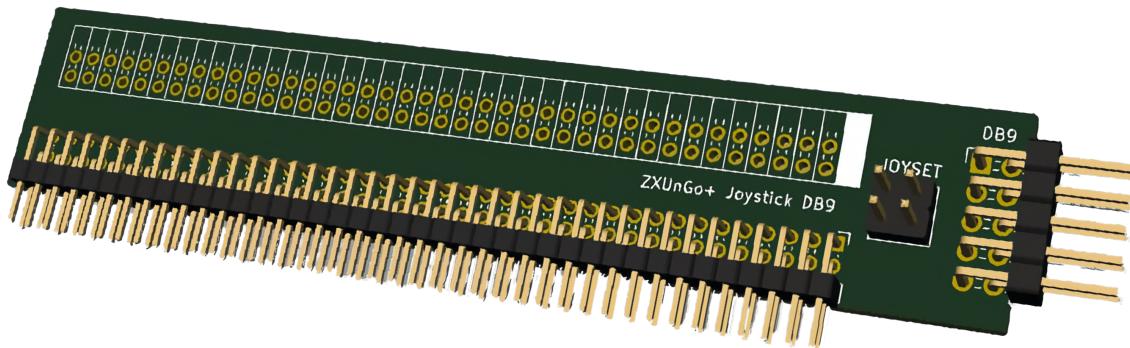


This is the module pinout (ZXUNO RGB666 / GRAPHICS GREMLIN):

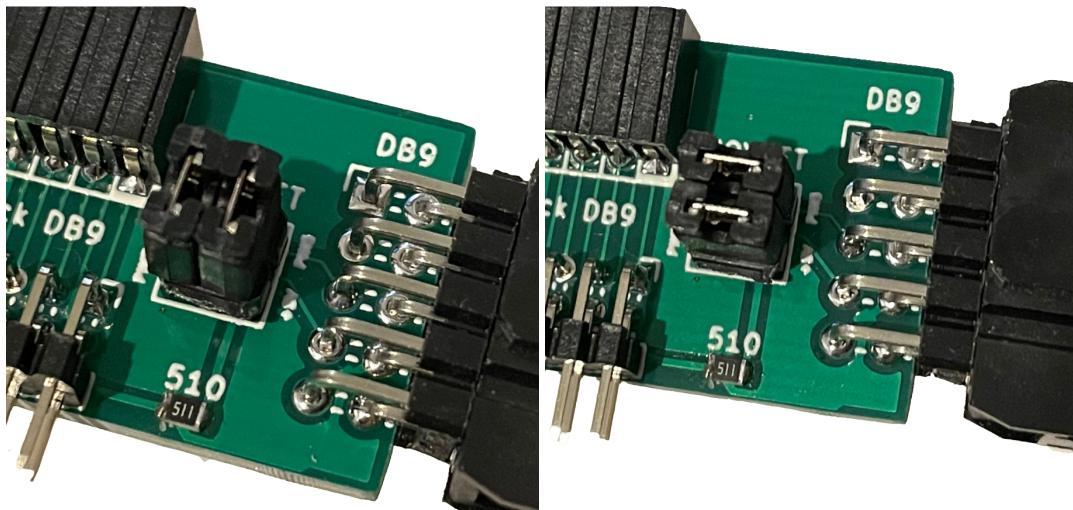
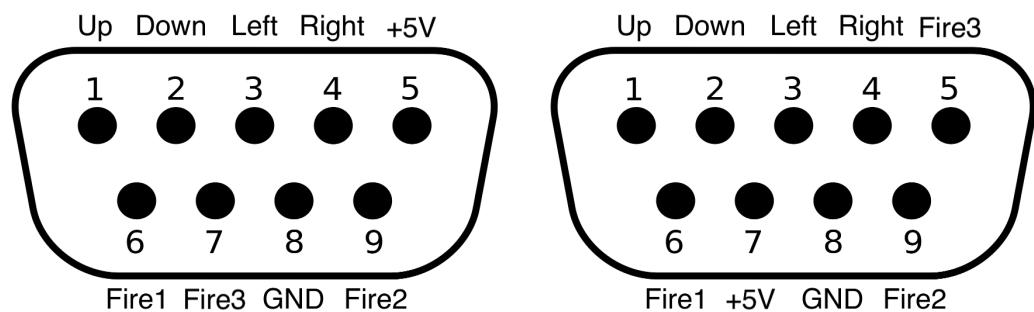


ZXUnGo+ DB9

Addon for ZXUnCore and ZXUnGo+ with DB9 Joystick connector (Atari standard), compatible with Antonio Villena's splitter.



There are configuration jumpers similar to those available on an original ZX-Uno. Thus, it is possible to configure **PIN5** to be **+5V** and **PIN7** to be **FIRE3** (left image below), or **PIN5** to be **FIRE3** and **PIN7** to be **+5V** (right image), which adds compatibility with Antonio Villena's splitter if desired.



The way to use it is to connect it with ZXUnCore and, if you have a ZXUnGo+, connect it behind. Once this is done, cores with joystick support will be able to use joysticks that connect to the DB9 port of the addon.



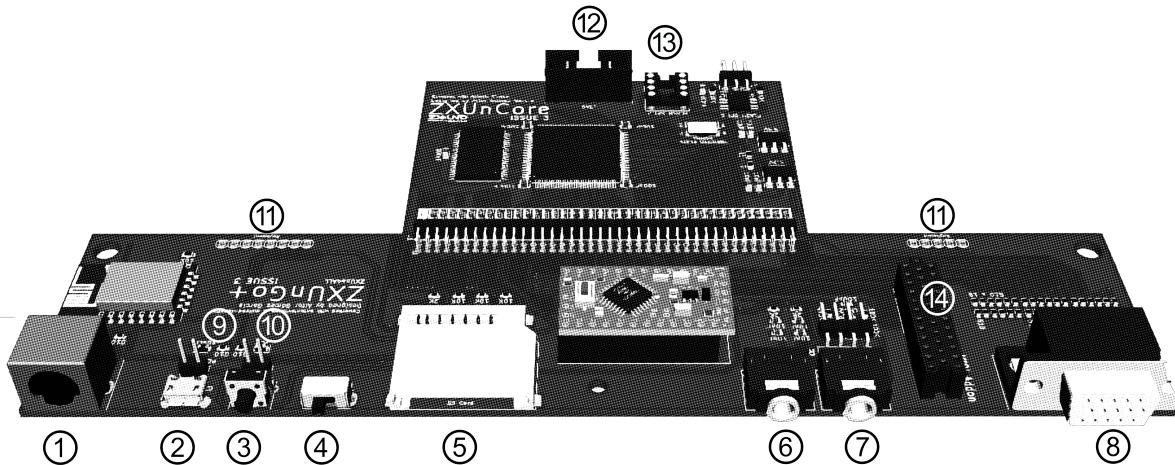
ZXUnGo+

ZX-Spectrum 16K/48K format board. A small case modification has to be made to get access to the VGA connector, by making larger the original TV output (if the screws and metal plate of the VGA connector aren't used, the enlargement is minimal). It's main features are:

- VGA video out
- SPI Flash selection switch
- Reset button
- New and improved EAR circuitry, designed by mcleod-ideafix, more sensitive when loading, enabling the load even from a mobile phone
- ZX-Spectrum membrane keyboard support, using ZXUnoPS2 and an Arduino Mini Pro based on Atmega 328P 5V version
- Wi-Fi module support
- MIDI addon connector (compatible McLeod MIDI addon)



Ports and Connectors



Description

1	Keyboard and mouse Port (PS/2)
2	Power (USB Mini-A)
3	Reset Button
4	SPI flash selection wtich
5	SD Card Slot
6	Audio Out
7	Audio In
8	VGA Video Out
9	Power button jumper
10	Reset button jumper
11	ZX-Spectrum membrane connectors
12	JTAG connector
13	External SPI Flash connector
14	MIDI module connector

Initial Setup

In order to be able to set up and use a ZXUnGo+ 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
- A ZX Spectrum (or ZX Spectrum+) membrane keyboard, or a PS/2 keyboard

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

- 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, a PC/Mac/Raspberry PI, etc. or a [cassette tape](#) recorder/player. The right sound channel is used as input (EAR) and the left channel can be used as output (MIC).

SD card formatting

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



FAT16 partitions have a maximum size of 4GB



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



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

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

Core	FAT 16	FAT 32	+3e	Primary Partition Type	Extra Partition	Access	Notes
ZX Spectrum EXP	Yes	Yes	Yes	Any	Yes	Full	Uses SPI Flash esxdos
ZX Spectrum Kyp 48K	Yes	Yes	No	Any	Yes	Full	Uses embedded esxdos
ZX Spectrum Kyp 128K	Yes	Yes	No	Any	Yes	Full	Uses embedded esxdos
ZX Spectrum TBBlue	Yes	Yes	No	Any	Yes	Full	Can use esxdos in SD
Acorn Atom	Yes	No	No	Any	No	Atom software archive	
Acorn Electron	Yes	No	No	Any	No	Only MMB file	
Amstrad CPC 464							Doesn't use the SD
Amstrad CPC 6128	No	Yes	No	0x0b (Win95 FAT-32)	No	Only disk image files .DSK in SD card root	4G or less partition and 4096 cluster
Apple I							Doesn't use the SD
Apple II	No	No	No	No	Exclusive	No	Exclusive special format
Arcades							SD not used
Atari 800XL	No	Yes	No	Any	No	Only ROM , disk or cartridge images	
Atari 2600	Si	No	No	Any	No	Only ROM files	
BBC Micro	Yes	Si	No	Any	No	Only MMFS or MMB file, depending on version	
Computers Lynx							Doesn't use the SD

Core	FAT 16	FAT 32	+3e	Primary Partition Type	Extra Partition	Access	Notes
CHIP-8	Yes	Yes	No	Any	No	Only ROMs (.BIN or .CH8)	
Colecovision	Yes	No	No	Any	No	Only ROMs (.ROM)	
Colour Genie							Doesn't use the SD
Commodore 16	No	No	No	No	Exclusive	No	Exclusive special format (C64 compatible)
Commodore 64	No	No	No	No	Exclusive	No	Exclusive special format (C16 compatible)
Commodore PET							Doesn't use the SD
Commodore VIC-20							Doesn't use the SD
Flappy Bird							Doesn't use the SD
Galaksija	Yes	No	No	Any	No	Only GTP files	
HT-1080Z	Yes	No	No	Any	No	Only `CAS` files	
Jupiter ACE	Yes	No	No	Any	No	Only TAP files	
MSX	Yes	No	No	0x06 (16-bit FAT)	Yes (FAT16)	Full	Not compatible with PC XT
NES	Yes	Yes	No	Any	No	Only ROMs (.NES)	
Ondra SPO 186							Doesn't use the SD
Oric Atmos	No	No	No	No	Exclusive	No	Exclusive special format
Oric Atmos (Kyp)							Doesn't use the SD
PC XT	Yes	No	No	0x06 (16-bit FAT)	No	Full. Needs DOS installed	Not compatible with MSX
PC XT CGA	Yes	Si	No	Depending on DOS version	No	Full. Needs DOS installed	Not compatible with MSX
Pong							Doesn't use the SD
SAM Coupé							Doesn't use the SD
Sega Master System	Yes	No	No	Any	No	Only SMS or BIN files	
SmartROM	Yes	Yes	No	Any	No	Needs specific files	
Vectrex	Yes	Yes	No	Any	No	Only BIN or VEC files	
Videopac	Yes	No	No	Any	No	Only ROMs (.BIN)	
ZX81							Doesn't use the SD
ZX81 and ZX80	Yes	Yes	No	Any	No	Only image files (.o or .p)	

Windows

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

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

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

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

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

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

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

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

macOS

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

In other case, you should use the command line.

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

```
diskutil unmountDisk /dev/disk6
diskutil partitionDisk /dev/disk6 MBR "MS-DOS FAT16" ZXUNCORE 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" ZXUNCORE 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% ZXUNCORE 4G %DOS_FAT_16% EXTRA 4G
"MS-DOS FAT32" DATA R
sudo newfs_msdos -F 16 -v ZXUNCORE -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" ZXUNCORE 4G "MS-DOS FAT32" EXTRA
R
```



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

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

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



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

Linux

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

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

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

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

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

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

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

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

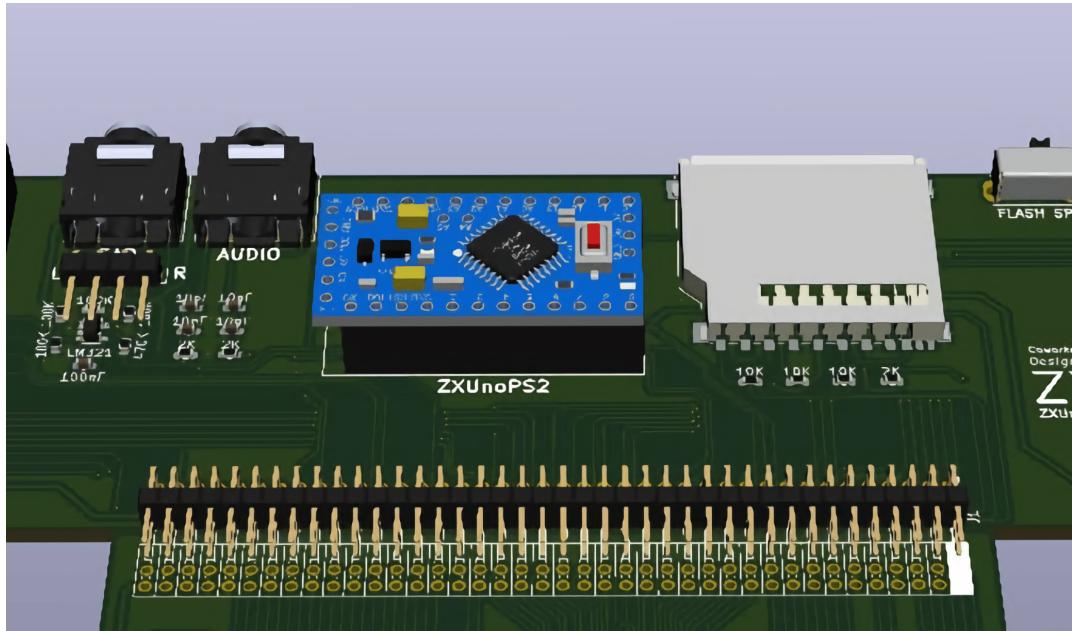
Format the FAT partition (requires root permissions)

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

Keyboard

Membrane keyboard

When using a ZXUnGo+, you can connect an original ZX Spectrum membrane keyboard. However, it lacks some of the existing keys on a modern PC keyboard. The keyboard membrane is connected to an Arduino board, which manages the transformation key presses to PS/2 keyboard protocol. The board is programmed so it can behave in different ways (keyboard modes) according to your needs.



Keyboard modes

The default is ZX Spectrum mode. To change to a different mode, you must press **Caps Shift+Symbol Shift+F**, then **U**, and then the key for the desired mode. After doing that, some text is automatically typed, to show the selected mode (for example **.zx** if you press **Caps Shift+Symbol Shift+F**, **U**, and then **1**).

Mode	Key
ZX Spectrum	0
Amstrad CPC	1
MSX	2
Commodore 64	3
Atari 800XL	4
BBC Micro	5
Acorn Electron	6
Apple (I and II)	7
Commodore VIC 20	8
PC XT	9
Oric Atmos	A
SAM Coupé	B
Jupiter ACE	C
Sega Master System	D
Commodore 16 / Plus 4	E
HT-1080Z / Video Genie / TRS-80 Model I	F
Galaksija	G
Multicomp	H

The ZX keyboard mode can behave in two different ways, which you can switch using **Caps Shift+Symbol Shift+F** and then **D**:

- Conventional mode, with access to the original **Caps Shift** and **Symbol Shift** but no access to special keys or modifier keys.
- Full mode, without access to **Caps Shift** or **Symbol Shift** but access to special and modifier keys just like the other keyboard modes.

In general, the keyboard layout (except on **ZX** conventional mode), showing the keypress obtained when combining **Caps Shift+Symbol Shift** (in full mode), is resumed here:

1	2	3	4	5	6	7	8	9	0
F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
Q	W	E	R	T	Y	U	I	O	P
F11	F12	*	-	Print Screen	Scroll Lock	Pause	Shift Lock	([and ^)	+ (* and])
A	S	D	F	G	H	J	K	L	Enter
Shift	< and >	+		Insert	Start	Page Up	Ñ	' ({ and ")	§ ({})
CShift	Z	X	C	V	B	N	M	SShift	Space
	Ctrl	Win	Alt	Del	End	Page Down	Alt Gr		



You don't have to stop pressing **Caps Shift+Symbol Shift** when combining modifier keys (e.g. for **Mayús.+F10** with **Caps Shift+Symbol Shift+A**, and **Caps Shift+Symbol Shift+0**).



Pressing **Caps Shift+Symbol Shift+F**, **U**, and an invalid key , types the name of the current mode.



The source code, binaries and more instructions about the keyboard firmware can be found at [the official repository](#).

Caps Shift+Symbol Shift+I behaves differently on the following cores:



- Commodore 16 and VIC-20: **Num Lock**
- Commodore Pet: **Menu** key (next to **Alt Gr**), used as **Shift Lock**



It's not recommended to use the **PC XT** mode, while also using a PS/2 keyboard

Functions

Pressing **Caps Shift+Symbol Shift+F** and then a special key, enables several special functions.

The full function list is this one:

Key	Action	Compatible Modes
V	Type Version	All
X	Save config to EEPROM	All
B	Ctrl+Alt+Bcksp (Hard Reset)	All
N	Ctrl+Alt+Del (Soft Reset)	All
Y	Ctrl+Alt+F5 (NMI)	ZX, CPC, Jupiter, SAM, ZX80, ZX81
O	F12 followed by Shift with a 1 sec. pause	BBC
H	F8+F10	Atari
U	Change keyboard mode	All
C	Map 0, P, Q, A keys to the keypad (Atari) or cursor keys	PC, MSX, SMS and Atari
D	Enable and disable full ZX keyboard mode	ZX
G	Change video mode	ZX, Atari, BBC, VIC, ORIC and HT-1080Z

C function enables or disables mapping the cursor keys to **0, P, Q, A** and **M** is exchanged with **Space**. It works with all keyboard modes except:

- C64 and Atari 800XL modes: instead of cursor keys, the map is to **8, 4, 6** and **2** of the keypad, and **M** to key **0** (fire)
- Jupiter ACE: instead of cursor keys, the map is to **8, 4, 6** and **2** of the keypad, and **M** and 'Space' are exchanged.

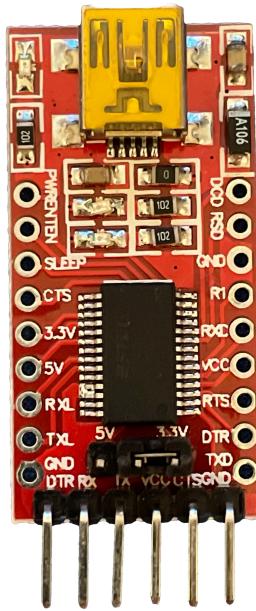
All keyboard modes try to match most of their symbols to those printed on the Spectrum keyboard. This way **Symbol Shift+M** obtains “.” or **Symbol Shift+K** obtains +. On some keyboards, a few symbols have been changed for more useful ones. For example, on version 1.3 spanish MSX, **Symbol Shift+E** obtains “í” and **Symbol Shift+I** obtains “¿”.

Firmware update

In order to install the firmware, a computer is needed (Windows, Mac, Linux).

Hardware needed:

- Computer (Windows, Mac, Linux)
- 6 [jump wires](#) (if possible, female on both sides)
- USB serial adapter (FTDI)

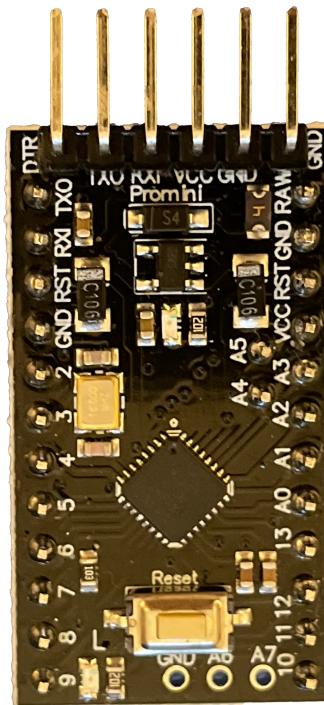


Software needed:

- [XLoader](#) (only for Windows) or [AVRDUDE](#) (you can use the embedded version in [Arduino IDE](#)). - .hex binary file for the firmware version to install, which you can download from [the official repository](#) (usually, the file is one whose name ends with [-328.hex](#)).

Preparation

Unplug the Arduino board (if needed), and locate all the connections.



Plug the serial adapter with the Arduino board, using this guide:

USB FTDI	Arduino Mini
DTR	DTR
RX	TXO
TX	RXI
VCC (3.3V)	VCC
CTS	GND
GND	GND



If the arduino board uses 5V for power, you can make also the connection using that instead of 3.3V

Plug the USB adapter to the computer, and find out the connected port ([COM](#), [/dev/usb…](#), etc.).

xLoader

Choose the following parameters:

- Device: Duemilanove/Nano (ATMega328)
- Baud rate: 57600
- Port: COM port for the adapter

Click "Upload" and wait until a message is shown saying "xx bytes uploaded".

AVRDUEDE

Use a command with this syntax:

```
avrdude -U flash:w:<firmware file.hex>:i -e -p atmega328p -b 57600 -c arduino -P <USB port>
```

When using the `avrdude` binary included with Arduino IDE, you also have to add the path to the included `.conf` file. For example, on MacOS:



```
/Applications/Arduino.app/Contents/Java/hardware/tools/avr/bin/avrdude      -U  
flash:w:zxunops2-25092021-328.hex:i -e -p atmega328p -b 57600 -c arduino -P  
/dev/cu.usbserial-A50285BI  
/Applications/Arduino.app/Contents/Java/hardware/tools/avr/etc/avrdude.conf      -C
```

PS/2 keyboard

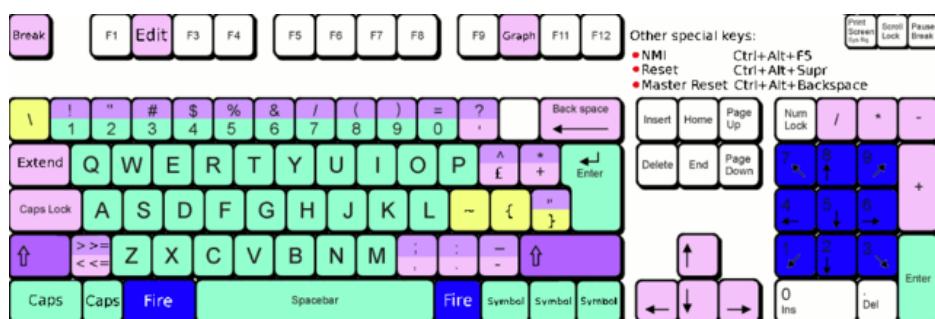
The map of a keyboard connected to the PS/2 port (physical keys of the keyboard assignment to the keystrokes that are presented to the different cores) is changed using the **Advanced** menu of the BIOS. There are three different maps to choose from: Spanish (default), English, and Spectrum (advanced).

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.

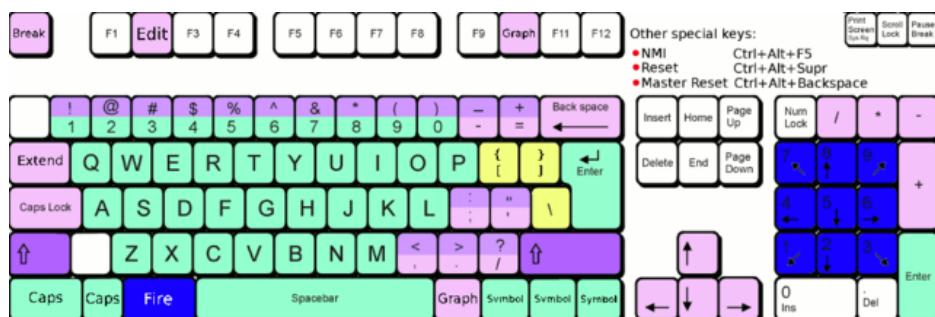
For the map to be preserved after a master reset, it has to be selected as **Default** in the BIOS.

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

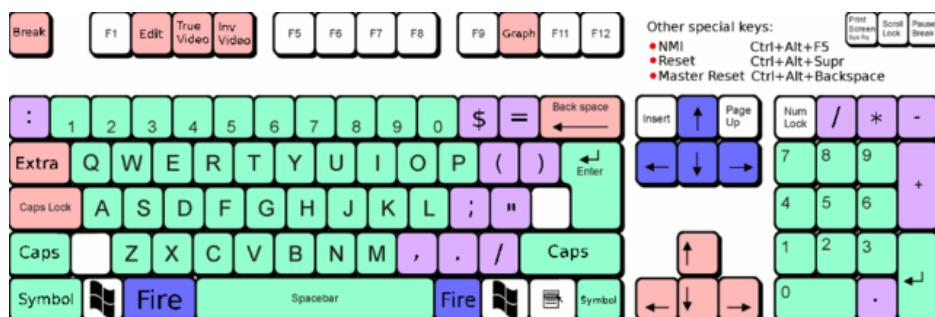
Spanish



English



Spectrum



Special keys and buttons

The following ZXUnGo+ key combinations are in **ZX** keyboard mode. Please check [the corresponding section](#) for more information.

Special keys which can be used during startup:

- **F2** (**Caps Shift+1** on ZXUnGo+) Enter BIOS setup
- **Caps Lock** or **Cursor down** (**Caps Shift+2** on ZXUnGo+) or, if a joystick is connected, pressing **down**: Core selection menu
- **Esc** (**Caps Shift+Space** on ZXUnGo+), 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, **Symbol Shift+V** on ZXUnGo+): 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** (**Caps Shift+Space** on ZXUnGo+): BREAK
- **F1** (**Caps Shift+Symbol Shift+1** on ZXUnGo+): (**Caps Shift+Symbol Shift+1** on ZXUnGo+): Select one of the monochrome color modes
- **F2** (**Caps Shift+1** on ZXUnGo+): Edit
- **F3** (**Caps Shift+3** on ZXUnGo+): True Video
- **F4** (**Caps Shift+4** on ZXUnGo+): Inverse Video
- **F5** (**Caps Shift+Symbol Shift+5** or **Caps Shift+Symbol Shift+Y** on ZXUnGo+): NMI
- **F6** (**Caps Shift+Symbol Shift+6** on ZXUnGo+): Sets a custom mark at the current position of a loaded PZX (only in the core version with PZX support)
- **F7** (**Caps Shift+Symbol Shift+7** on ZXUnGo+): Play or pause when playing .PZX files
- **F8** (**Caps Shift+Symbol Shift+8** on ZXUnGo+): Rewind .PZX file to the previous mark
- **CTRL+ F8** (**Caps Shift+Symbol Shift+Z**, and then, **Caps Shift+Symbol Shift+8** on ZXUnGo+): Rewind PZX to a "tag BRWS" mark. These marks can be used, for example, to point to the start of a multiload (only in the core version with PZX support)
- **F9** (**Caps Shift+Symbol Shift+9** on ZXUnGo+): Stop playing a PZX and rewind to the start
- **F10** (**Caps Shift+9** on ZXUnGo+): Graph
- **F12** (**Caps Shift+Symbol Shift+W** on ZXUnGo+): Turbo Boost. Switches the CPU speed between 28MHz and 3.5MHz.
- **Ctrl+Alt+Backspace** (**Caps Shift+Symbol Shift+B** on ZXUnGo+): Hard reset. Backspace is the delete key, located in the top-right portion of the keyboard, above **Enter**.
- **Ctrl+Alt+Supr** (**Caps Shift+Symbol Shift+N** on ZXUnGo+): Soft reset.
- **End** (**Caps Shift+Symbol Shift+B** on ZXUnGo+): Switch between colour, green, amber and black and white modes

esxdos

esxdos is a firmware for the DivIDE/DivMMC hardware interfaces (which ZX-Uno 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 latest version is 0.8.9, 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 ZX-Uno Spectrum core, you will see how esxdos detects the card and loads the necessary components to work.

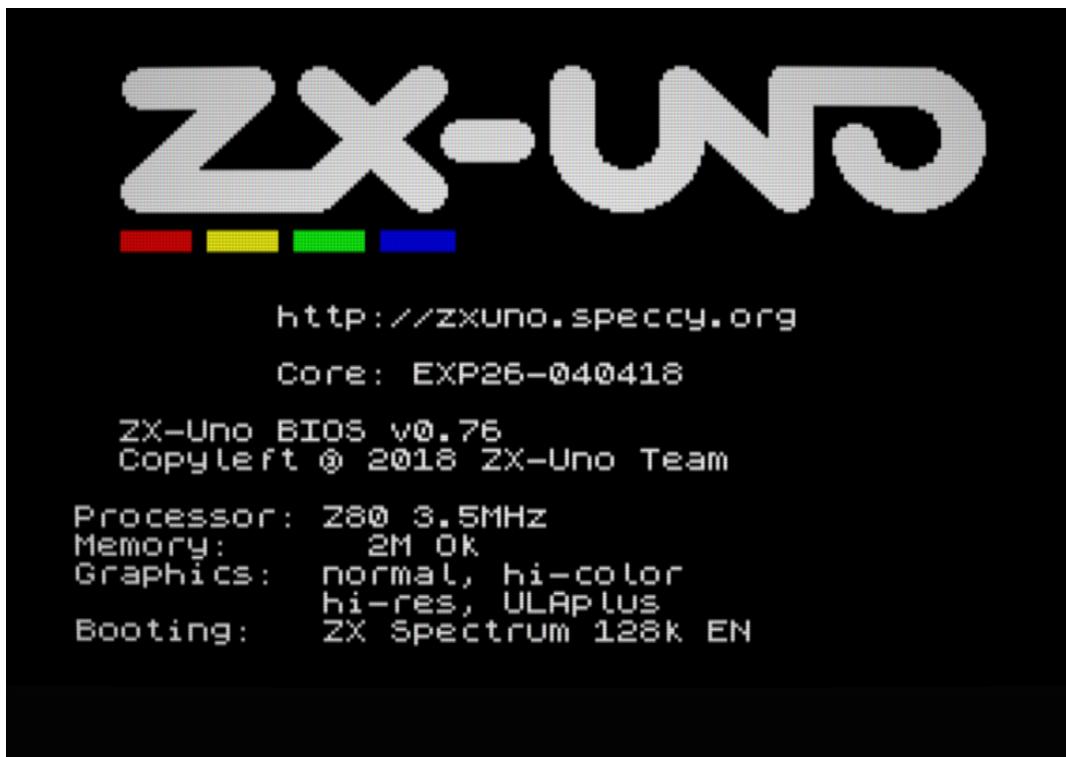


It is also recommended to add the specific esxdos commands for ZX-Uno. These can be obtained from the project source page ([here](#), [here](#) and [here](#)), and are as follows:

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

It is [explained later](#) what each of them does.

BIOS



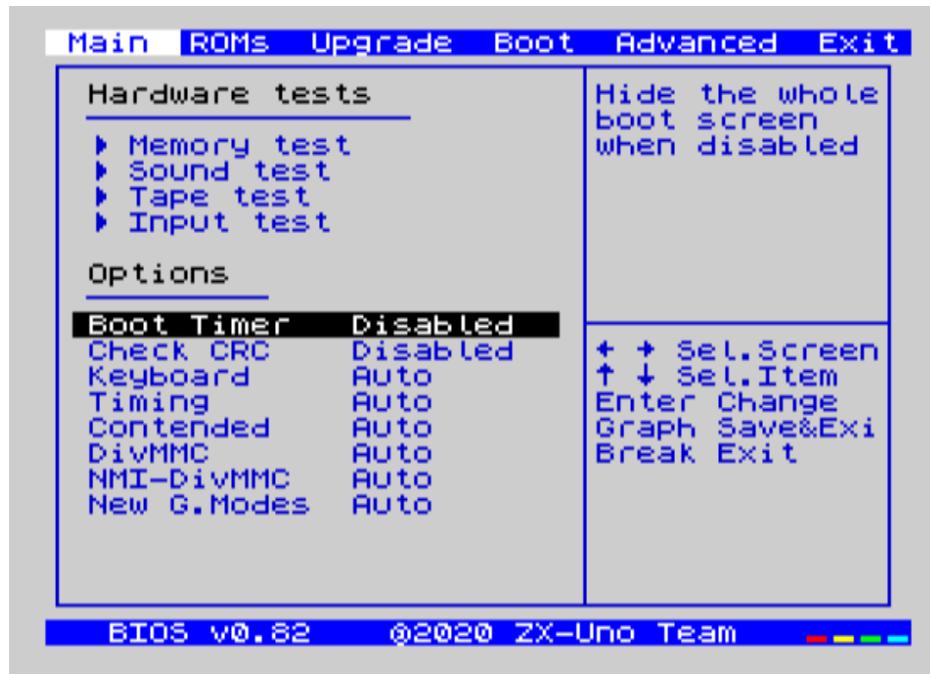
Pressing the **F2** (**Caps Shift+1** on ZXUnGo+) key during boot will access the BIOS setup. The BIOS firmware is the first program that runs when the ZX-Uno is turned on. The main purpose of BIOS is to start and test the hardware and load one of the installed cores.

Using left and right cursor keys (**Caps Shift+5** and **Caps Shift+8** on ZXUnGo+), you can navigate through the BIOS setup screens. With up and down keys **Caps Shift+7** and **Caps Shift+6** on ZXUnGo+) 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 (**Caps Shift+Space** on ZXUnGo+) is used to close open option windows without applying any action.

Other useful keys to use on startup are:

- **Scroll Lock** or **Cursor Down** or **Caps Shift+6** or **Caps Shift+2**, or, if a joystick is connected, pressing the down direction button: Core Selection Menu
- **Esc** or **Caps Shift+Space**, or, if a joystick with two or more buttons is connected, pressing the second fire: ZX Spectrum Core ROM selection menu
- **R**: Load the ZX Spectrum default ROM in "real" mode disabling esxdos, new graphic modes, etc. (combined with **Esc** let's you use a different ROM)
- **/** (numeric keypad) or **Symbol Shift+V**: Load the ZX Spectrum default ROM in "root" mode (combined with **Esc** let's you use a different ROM)
- Number between **1** and **9**: Load the corresponding core from Flash

Main

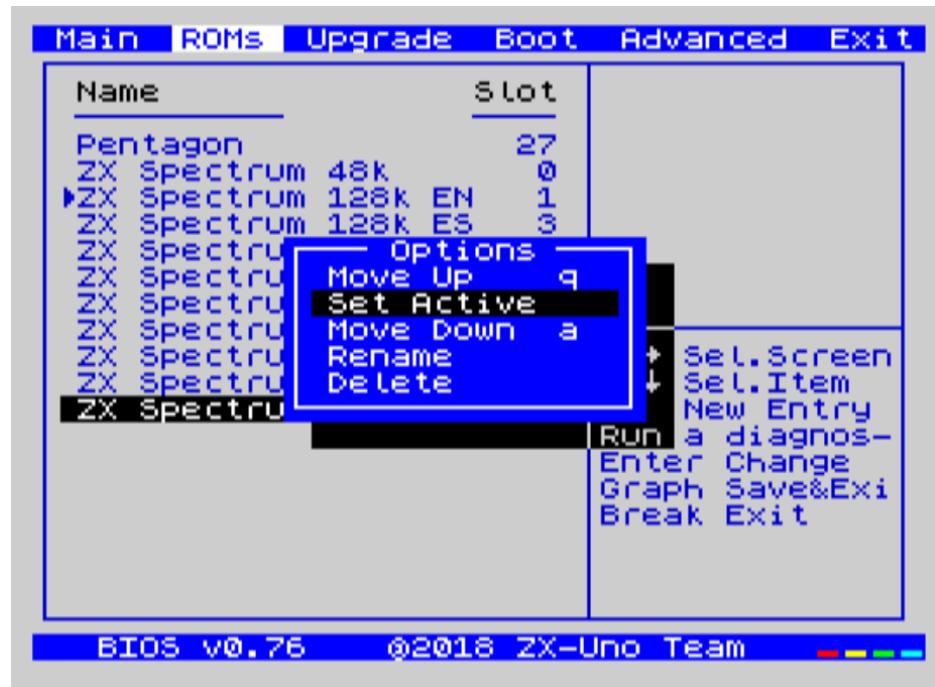


In the first configuration screen, in addition to being able to run several tests, you can define the default behavior for the following:

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

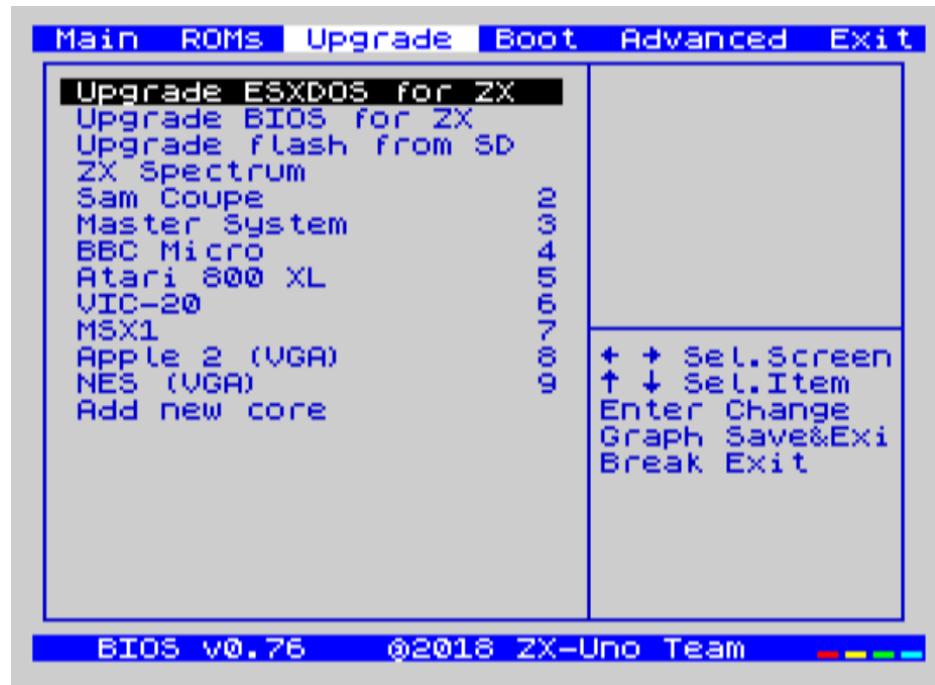
More technical information can be found on [the ZX-Uno Wiki](#).

ROMs



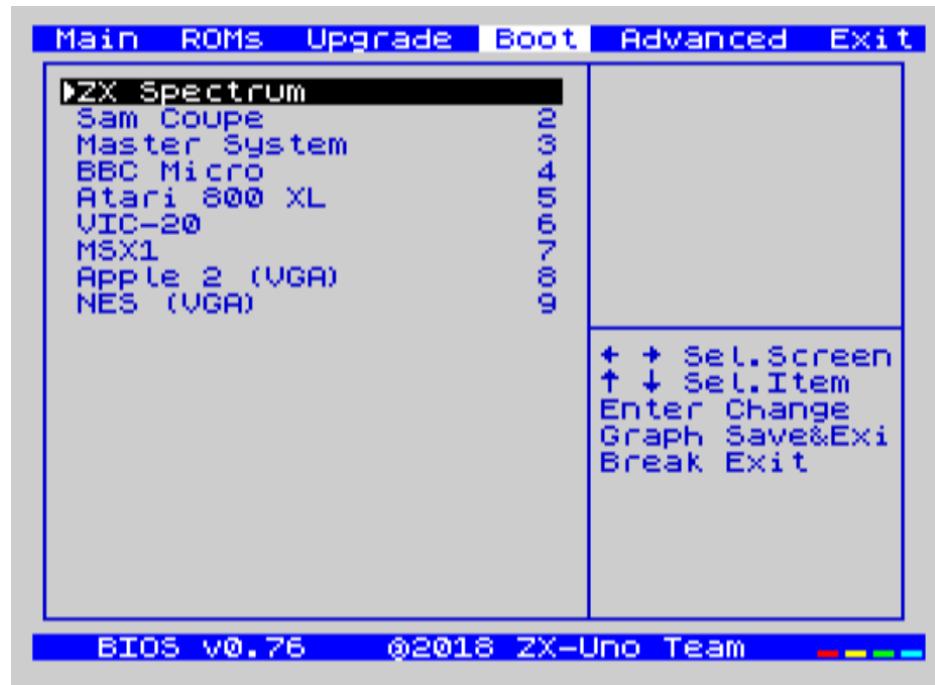
The second screen shows the installed ZX Spectrum ROMs. You can reorder (Move Up, Move Down), rename or delete each of them, as well as choose the one that will be loaded by default at startup (Set Active).

Upgrade



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

Boot

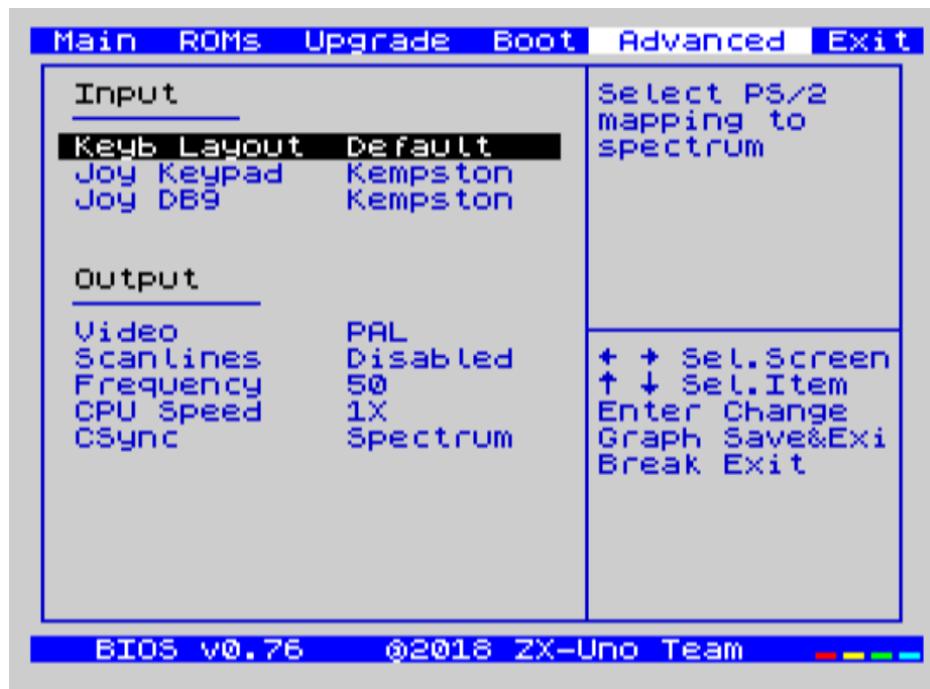


In the *Boot* screen you can choose which one of the installed cores is loaded by default at startup.



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

Advanced



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

- Keyboard layout (Keyb Layout): See [the corresponding section](#) for more information)
- Joystick behavior when emulated with the numeric keypad (Joy Keypad) and the second joystick (with ZXUnGo+ DB9 and a splitter): Kempston, Sinclair Joystick 1, Sinclair Joystick 2, Protek or Fuller
- Behavior of a joystick connected to the main port of a ZXUnGo+ DB9 (Joy DB9): Kempston, Sinclair Joystick 1, Sinclair Joystick 2, Protek, Fuller or simulate the keys **Q**, **A**, **0**, **P**, **Space** and **M**
- Enable the second physical joystick (ZXUnGo+ DB9 and splitter). Only on ZXUnGo+ DB9 with a splitter connected and for cores which have splitter support
- Video output: PAL, NTSC or VGA
- Scanlines simulation: Enabled or Disabled
- VGA horizontal frequency: 50, 51, etc.
- CPU speed: Normal (1x) or accelerated (2X, 3X, etc.)
- Csync: Spectrum or PAL

This table explains how the joystick buttons are assigned:

Button	Kempston	Sinclair 1	Sinclair 2	Protek	Fuller	QAOPSPcM
Left	Bit 0	1	6	5	Bit 2	0
Right	Bit 1	2	7	8	Bit 3	P
Down	Bit 2	3	8	6	Bit 1	A
Up	Bit 3	4	9	7	Bit 0	Q

Button	Kempston	Sinclair 1	Sinclair 2	Protek	Fuller	QAOPSpCM
Fire 1	Bit 4	5	0	0	Bit 4	Space
Fire 2	Bit 5	X	Z	9	Bit 5	M

Exit



Finally, from the last screen you can:

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

ZX Spectrum

The main core is the one implementing a ZX Spectrum computer. This core is special, and it cannot be replaced for another that is not a ZX Spectrum, since the ZX-Uno uses it for its operation. The one installed by default is a variant of EXP27-160820 mcleod_ideafix core.

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
- Covox support
- Each channel A, B, C of the two AY-3-8912, beeper and SpecDrum chips can be directed to the left, right, both or neither outputs, allowing the implementation of configurations such as ACB, ABC, etc.
- Real joystick and keyboard joystick support with Kempston, Sinclair 1 and 2, Cursor, Fuller and QAOPSpcM protocol.
- Turbo mode support at 7MHz, 14MHz, 28MHz
- Keyboard support (PS/2 protocol) and user-configurable mapping from within Spectrum itself.
- PS/2 mouse support emulating the Kempston Mouse protocol.
- Possibility of video output in composite video mode, RGB 15kHz (PAL and NTSC), or VGA.
- User selectable vertical refresh rate to improve compatibility with VGA monitors.
- Multicore boot support: from the Spectrum you can select an address of the SPI Flash and the FPGA will load a core from there.
- Wi-Fi support (UART) via addon

- MIDI support (General MIDI) via addon
- Loading **PZX** tape files from SD

The version included with ZXUnCore has the following enhancements:

- RGB66 video DAC support
- Color, green, amber and black and white modes pressing the **End** key (**Caps Shift+Symbol Shift+B** on ZXUnGo+)
- Turbo Boost Mode: The **F12** key (**Caps Shift+Symbol Shift+W** on ZXUnGo+) switches the CPU speed between 28Mhz (4x) and 3,5Mhz (1x).
- Loagint .PZX has been enhanced, getting back the option to have a custom mark

As it could be done with the first unofficial T24 core synthesized by Neuro when loading .PZX files, the following keys can be used:

- **F6** (**Caps Shift+Symbol Shift+6** on ZXUnGo+): Sets a custom mark at the current position of a loaded PZX (only in the core version with PZX support)
- **F7** (**Caps Shift+Symbol Shift+7** on ZXUnGo+): Play or pause when playing .PZX files
- **F8** (**Caps Shift+Symbol Shift+8** on ZXUnGo+): Rewind .PZX file to the previous mark
- **CTRL+ F8** (**Caps Shift+Symbol Shift+Z**, and then, **Caps Shift+Symbol Shift+8** on ZXUnGo+): Rewind PZX to a "tag BRWS" mark. These marks can be used, for example, to point to the start of a multiload (only in the core version with PZX support)
- **F9** (**Caps Shift+Symbol Shift+9** on ZXUnGo+): Stop playing a PZX and rewind to the start

When a PZX file is fully played, the player stops automatically. It also stops if a STOP mark is found or if a STOP IF IN 48K is found and the ZX-Uno is in 48K mode (no 128K pagination).



You can add BRWS or STOP marks to a PZX file with the program **ZX-Blockeditor**, which belong to the set of **ZX-Modules** utilities.

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 ZX-Uno, and you can choose which one to load by pressing the **Esc** key during boot. You can also define the ROM that you want to load by default using the BIOS setup.

See the [updates](#) section for more information on how to expand or modify the ROMs stored in flash memory.

DerbyPro

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



Features include:

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

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

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

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

CargandoLeches

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

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

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

The whole ROM set is available to download from the repository in GitHub [here](#).

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

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

POKEs

When using a ROM with POKE option enabled:

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

Preparing ultrafast loading tapes

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

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

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

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

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

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

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

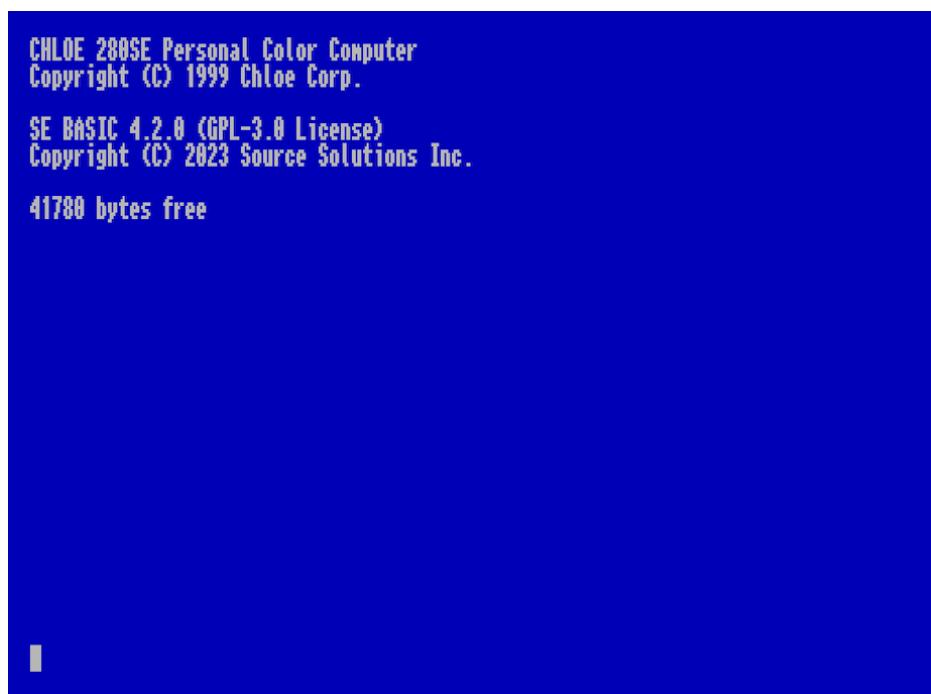
This way, the file **Valley.wav** can be played from a computer or another device and load using the ROM (see the section about [loading from tape](#) for more info).

SE Basic IV

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

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

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



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

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

Features include:

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



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



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

Other ROMs

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

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

SD advanced format (+3e)

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

+3e uses its own partition format (called IDEDOS), to split 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 ZXUNCORE -c 128 /dev/rdisk6s2
sudo newfs_msdos -F 32 -v EXTRA -c 128 /dev/rdisk6s3
```

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

```
diskutil list
```

```
(...)
/dev/disk6 (external, physical):
 #:          TYPE NAME      SIZE    IDENTIFIER
 0:  FDisk_partition_scheme          *15.9 GB   disk6
 1:          0x7F                 1.0 GB   disk6s1
 2:  DOS_FAT_16 ZXUNCORE        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 ZXUNCORE -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 ZX-Uno, 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](#).

esxdos Commands

Basic Guide

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

The main "DOT" commands are the following:

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



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

ZX-Uno commands

As explained in the installation part, there are a series of commands that are exclusive to ZX-Uno:

- **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

Wi-Fi

An ESP module with a Wi-Fi chip [ESP8266](#) can be added, that can be easily used with a ZX Spectrum core (e.g., EXP27 160820 core) which has synthesized an [UART](#) device, that allows communication with the module.

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

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

For example:

```
.iwconfig mywifi mypassword
```



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

Network tools for ZX-Uno pack

These are programs, developed by Nihilash and that are available to [download from his web](#).

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

FTP-Uno

FTP cliente developed by Yombo, available [at GitHub](#).

Configuration steps:

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

UART Terminal

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

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

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

You can see all the available commands reading the [official documentation](#).

Making RDM (RaDastan Movie) files

The `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](#).

FUZIX

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

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

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

How to Build

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

Install the needed packages:

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

Get the special version of [SDCC compiler](#) for Fuzix, and build it:

```
git clone https://github.com/EtchedPixels/sdcc280.git  
  
cd sdcc280  
cd sdcc  
.configure  
make  
sudo make install  
cd ../../
```

Get Fuzix source code:

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

Edit [Makefile](#) and change the line with TARGET= to TARGET=zxdiv. Build:

```
sudo make
```

Get the esxdos binary and kernel image from this paths:

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

Build the root filesystem:

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

Get the root filesystem image file from this location:

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

How to use

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

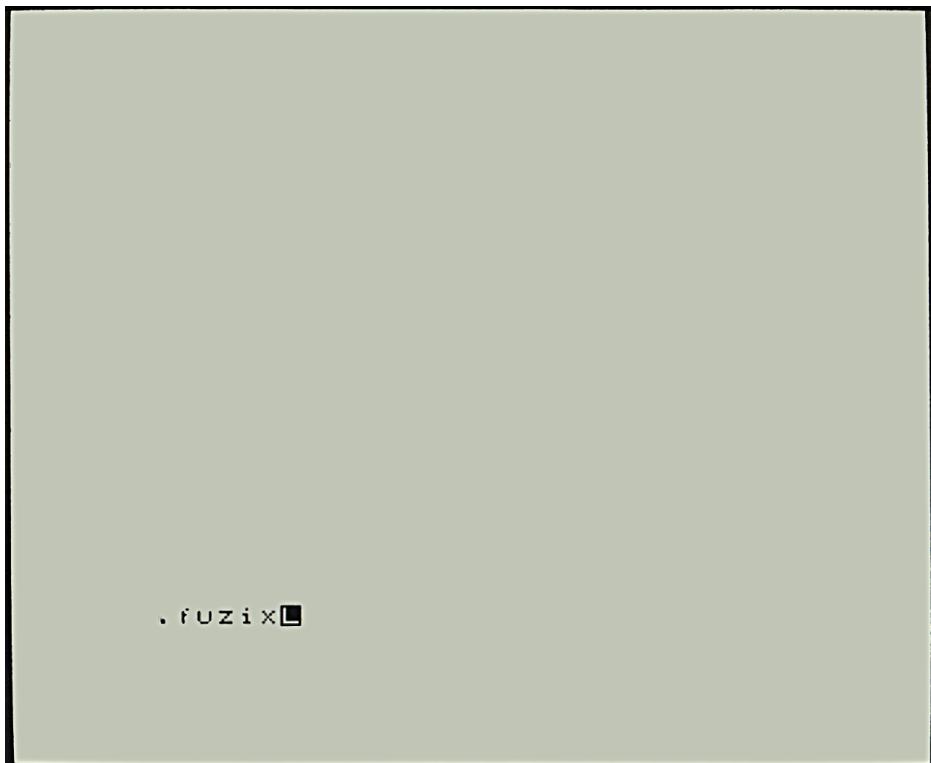
Copy the `rootfs` filesystem to the type [7E](#) partition. You can use the `dd` utility, included with Linux, macOS, etc. (and also [ported to Windows](#)).

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

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

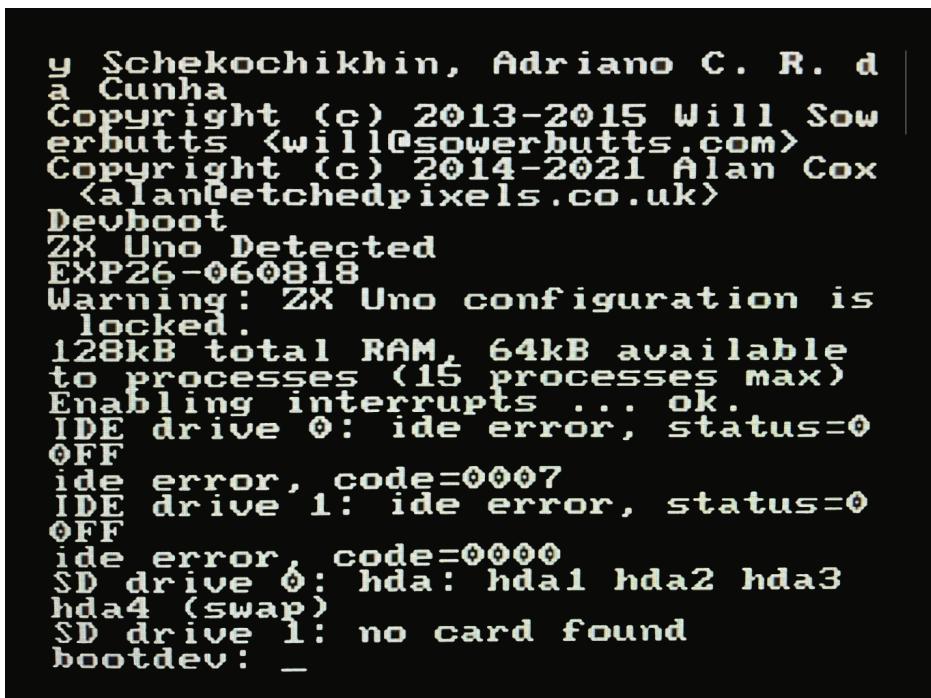
Copy the `FUZIX` command into the `BIN` directory and copy `FUZIX.BIN` to the top level directory of the esxdos partition.

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



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

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



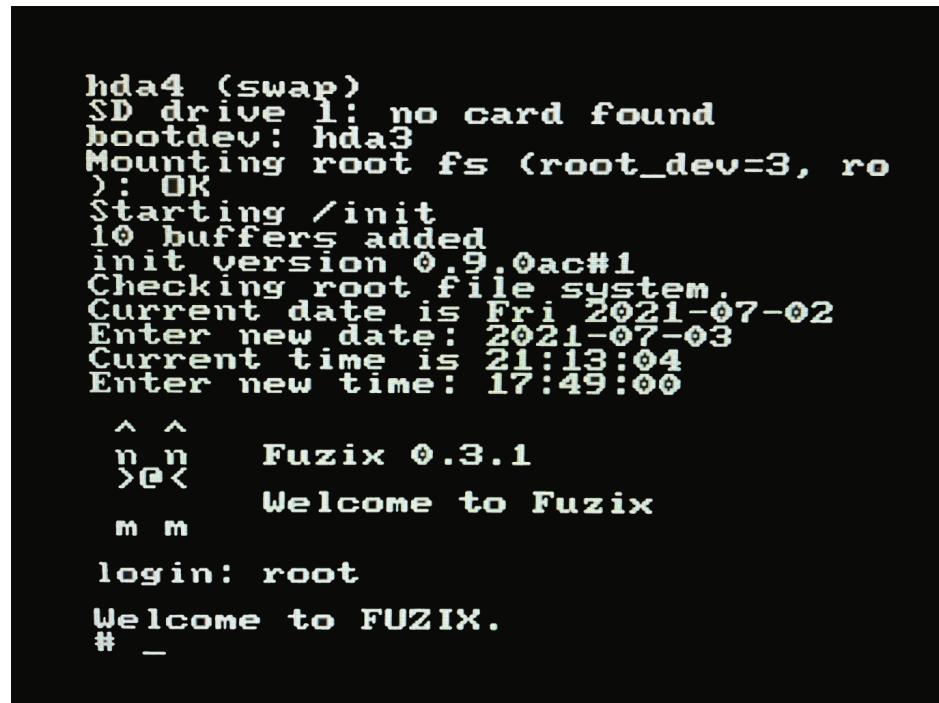
Type the rootfs partition (e.g. `hd3`) and press **Enter**.

```
Devboot
ZX Uno Detected
EXP26-060818
Warning: ZX Uno configuration is
locked.
128kB total RAM, 64kB available
to processes (15 processes max)
Enabling interrupts ..: ok.
IDE drive 0: ide error; status=0
OFF
ide error, code=0007
IDE drive 1: ide error, status=0
OFF
ide error, code=0000
SD drive 0: hda: hda1 hda2 hda3
hda4 (swap)
SD drive 1: no card found
bootdev: hda3
Mounting root fs (root_dev=3, ro
): OK
Starting /init
10 buffers added
init version 0.9.0ac#1
-
```

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

```
locked.
128kB total RAM, 64kB available
to processes (15 processes max)
Enabling interrupts ..: ok.
IDE drive 0: ide error; status=0
OFF
ide error, code=0007
IDE drive 1: ide error, status=0
OFF
ide error, code=0000
SD drive 0: hda: hda1 hda2 hda3
hda4 (swap)
SD drive 1: no card found
bootdev: hda3
Mounting root fs (root_dev=3, ro
): OK
Starting /init
10 buffers added
init version 0.9.0ac#1
Checking root file system.
Current date is Fri 2021-07-02
Enter new date: 2021-07-03
Current time is 21:13:04
Enter new time: 17:49:00_
```

Login with `root` user and no password.



```
hda4 (swap)
SD drive 1: no card found
bootdev: hda3
Mounting root fs (root_dev=3, ro
): OK
Starting /init
10 buffers added
init version 0.9.0ac#1
Checking root file system.
Current date is Fri 2021-07-02
Enter new date: 2021-07-03
Current time is 21:13:04
Enter new time: 17:49:00

^ ^
n n   Fuzix 0.3.1
>@<   Welcome to Fuzix
m m

login: root
Welcome to FUZIX.
# _
```

Now you have a Fuzix shell.



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

Upgrade

BIOS

To update the BIOS, a file named **FIRMWARE.ZX1** 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 ZX-Uno 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 ZX-Uno 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 ZX-Uno using a "rooted" ROM, and then just enter the command **.romsupgr**. This will burn all the ROMs, which will be available for use.



Remember that [if the ZX-Uno is started while pressing the / key](#), then the default ROM of the ZX Spectrum core will be loaded in "root" mode.

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

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

Cores

There are 45 available spaces where you can store cores, 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 ZX-Uno commands).

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

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

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

If everything has been done correctly, when you turn on the ZX-Uno 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 ZX-Uno and press the **F2** key during boot to access the BIOS setup. Select the menu **Upgrade** and then choos the option "*Upgrade flash from SD*". Press Enter, choose **Yes**, and press Enter again to start the Flash writing process.

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



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

Other cores

ZX Spectrum EXP27 without PZX

This core is like the default one, but adding a SAA1099 audio chip and without PZX support.

ZX Spectrum 48K (Kyp)

[Alternative core](#), whose objective is to be the most accurate implementation in timings, memory contention, etc.

Main features:

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

SD card format

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

See the [corresponding section](#) for instructions of how to install this alternative core in ZX-Uno.

Keyboard

Special keys and buttons

The following ZXUnGo+ keyboard combinations are obtained while in **ZX** mode. Please read [the section dedicated to ZXUnGo+ keyboard modes](#) for more information.

While the core is running:

- **Esc** (**Caps Shift+Space** on ZXUnGo+): BREAK
- **F5** (**Caps Shift+Symbol Shift+5** or **Caps Shift+Symbol Shift+Y** on ZXUnGo+): NMI
- **F8** (**Caps Shift+Symbol Shift+8** on ZXUnGo+): Change Turbosound mixer configuration between ACB and ABC.
- **Ctrl+Alt+Backspace** (**Caps Shift+Symbol Shift+F** and then **B** on ZXUnGo+): Hard reset. Backspace is the delete key, located in the top-right portion of the keyboard, above **Enter**.
- **Ctrl+Alt+Supr** (**Caps Shift+Symbol Shift+F** and then **N** on ZXUnGo+) o **F12** (**Caps Shift+Symbol Shift+W** on ZXUnGo+): Soft reset.

ZX Spectrum 128K (Kyp)

[Alternative core](#), whose objective is to be the most accurate implementation in timings, memory contention, etc.

Main features:

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

SD card format

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

See the [corresponding section](#) for instructions of how to install this alternative core in ZX-Uno.

Keyboard

Special keys and buttons

The following ZXUnGo+ keyboard combinations are obtained while in **ZX** mode. Please read [the section dedicated to ZXUnGo+ keyboard modes](#) for more information.

While the core is running:

- **Esc** ([Caps Shift+Space](#) on ZXUnGo+): BREAK
- **F5** ([Caps Shift+Symbol Shift+5](#) or [Caps Shift+Symbol Shift+Y](#) on ZXUnGo+): NMI
- **F8** ([Caps Shift+Symbol Shift+8](#) on ZXUnGo+): Change Turbosound mixer configuration between ACB and ABC.
- **Ctrl+Alt+Backspace** ([Caps Shift+Symbol Shift+F](#) and then **B** on ZXUnGo+): Hard reset. Backspace is the delete key, located in the top-right portion of the keyboard, above **Enter**.
- **Ctrl+Alt+Supr** ([Caps Shift+Symbol Shift+F](#) and then **N** on ZXUnGo+) or **F12** ([Caps Shift+Symbol Shift+W](#) on ZXUnGo+): Soft reset.

ZX Spectrum (Kyp/azesmbog)

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

Its main features are:

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

SD card format

You need a SD card with the first partition formatted as FAT16 or FAT32, and inside, the standard esxDOS 0.8.9 (see [esxdos corresponding section](#) for more info).

See the [corresponding section](#) for instructions of how to install this alternative core in ZX-Uno.

Keyboard

Special keys and buttons

The following ZXUnGo+ keyboard combinations are obtained while in [ZX](#) mode. Please read [the section dedicated to ZXUnGo+ keyboard modes](#) for more information.

While the core is running:

- [F5 \(Caps Shift+Symbol Shift+5 or Caps Shift+Symbol Shift+Y on ZXUnGo+\)](#): NMI
- [F8 \(Caps Shift+Symbol Shift+8 on ZXUnGo+\)](#): Change Turbosound between ACB and ABC.
- [F10 \(Caps Shift+Symbol Shift+0 on ZXUnGo+\)](#): Pentagon Mode
- [F11 \(Caps Shift+Symbol Shift+Q on ZXUnGo+\)](#): 48K Mode
- [F12 \(Caps Shift+Symbol Shift+W on ZXUnGo+\)](#): 128K Mode
- [Ctrl+Alt+Backspace \(Caps Shift+Symbol Shift+F and then B on ZXUnGo+\)](#): Hard reset. Backspace is the delete key, located in the top-right portion of the keyboard, above [Enter](#).
- [Ctrl+Alt+Supr \(Caps Shift+Symbol Shift+F and then N on ZXUnGo+\)](#) or [F12 \(Caps Shift+Symbol Shift+W on ZXUnGo+\)](#): Soft reset.

The default mode on startup is 128K.



To see the standard menu on 128K mode, you have to use the esxdos dot command: [.128](#).

ZX Spectrum TBBlue

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

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

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

SD card format

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

Obtain the TBBlue distribution that matches the core version, for example, looking in [ZX-Uno forums](#). Usually, the structure is something like this:

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



If you are using a SD card with [esxdos](#), the [esxmmc.rom](#) file version must match the one of the SD card.

See the [corresponding section](#) for instructions of how to install the TBBBlue core in ZX-Uno.

Keyboard

Special keys and buttons

The following ZXUnGo+ keyboard combinations are obtained while in **ZX** mode. Please read [the section dedicated to ZXUnGo+ keyboard modes](#) for more information.

While the core is running:

- **F1** (**Caps Shift+Symbol Shift+1** on ZXUnGo+): Hard Reset
- **F1** (**Caps Shift+Symbol Shift+1** on ZXUnGo+) and then, **Space**: Boot menu
- **F2** (**Caps Shift+Symbol Shift+2** on ZXUnGo+): Enable or disable Scandoubler
- **F3** (**Caps Shift+Symbol Shift+3** on ZXUnGo+): Switches between 50Hz and 60Hz
- **F4** (**Caps Shift+Symbol Shift+4** on ZXUnGo+): Soft reset
- **F5** (**Caps Shift+Symbol Shift+5** on ZXUnGo+): CPU Pause
- **F9** (**Caps Shift+Symbol Shift+9** on ZXUnGo+): Multiface
- **F10** (**Caps Shift+Symbol Shift+0** on ZXUnGo+): NMI

Basic Guide

Pressing **F1** (**Caps Shift+Symbol Shift+1** on ZXUnGo+) and then, **Space**, forces a core reset and then the boot menu appears, where the existing boot options in the **config.ini** file are shown.



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



Acorn Atom

Acorn Atom was a home computer made by Acorn Computers Ltd. The ZX-Uno 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

The following ZXUnGo+ keyboard combinations are obtained while in **ATOM** mode (**Caps Shift+Symbol Shift+F**, then **U** and then **J**). Please read [the section dedicated to ZXUnGo+ keyboard modes](#) for more information.

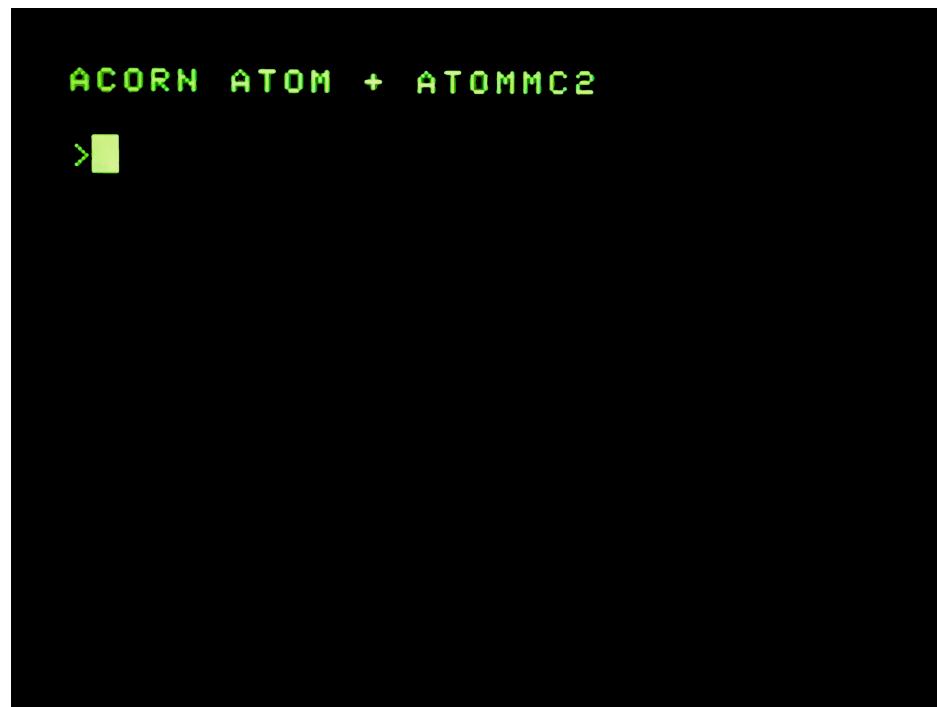
While the core is running:

- **Shift+F10** (**Caps Shift+Symbol Shift+A**, and then **Caps Shift+Symbol Shift+0** on ZXUnGo+): Shows Atom Software Archive Menu
- **F10** (**Caps Shift+Symbol Shift+0** on ZXUnGo+): Soft Reset
- **F1** (**Caps Shift+Symbol Shift+1** on ZXUnGo+): Turbo mode 1Mhz
- **F2** (**Caps Shift+Symbol Shift+2** on ZXUnGo+): Turbo mode 2Mhz
- **F3** (**Caps Shift+Symbol Shift+3** on ZXUnGo+): Turbo mode 4Mhz
- **F4** (**Caps Shift+Symbol Shift+4** on ZXUnGo+): Turbo mode 8Mhz

The keyboard uses the following mapping:



Basic Guide



Once it's running, press **Shift+F10** (**Caps Shift+Symbol Shift+A**, and then **Caps Shift+Symbol Shift+0** on ZXUnGo+), or type ***MENU** and **Enter**, to show a menu where you can choose and load Atom Software Archive programs from the card.

Acorn Electron

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

Main features:

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

SD card format

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

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

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

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



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

Keyboard

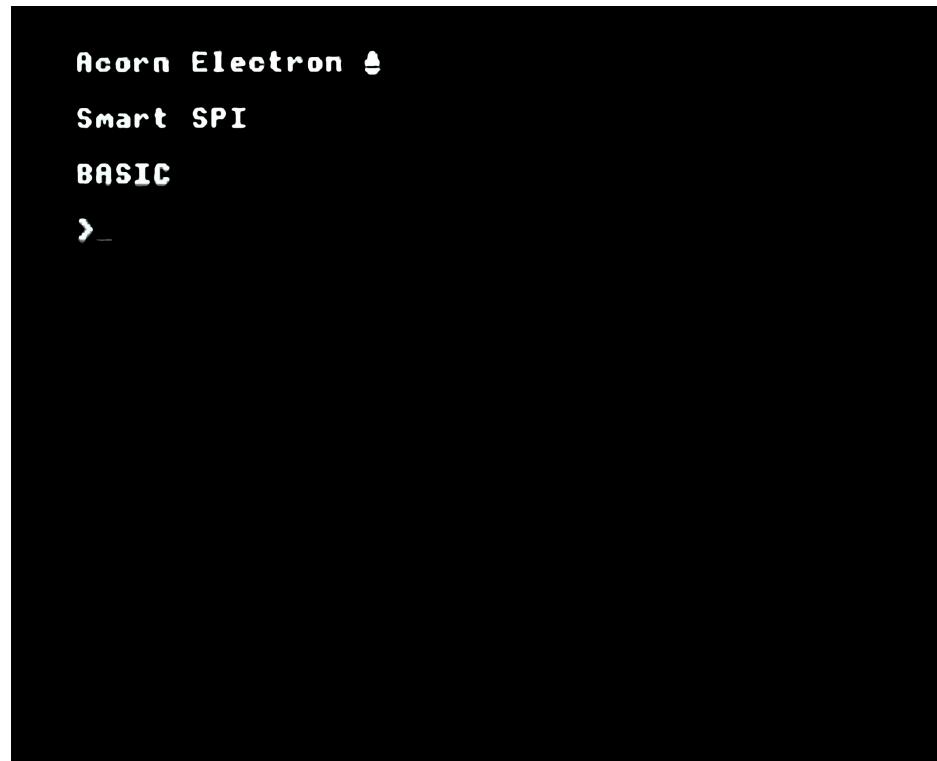
Special keys and buttons

The following ZXUnGo+ keyboard combinations are obtained while in ELECTRON mode ([Caps Shift+Symbol Shift+F](#), then [U](#) and then [7](#)). Please read [the section dedicated to ZXUnGo+ keyboard modes](#) for more information.

While the core is running:

- [Ctrl+Shift Lock+1](#) to [Ctrl+Shift Lock+4](#) (external PS/2 keyboard required): Use other graphic modes (60Hz, etc)
- [F10](#) ([Caps Shift+Symbol Shift+0](#) on ZXUnGo+) and [Ctrl+F10](#) ([Caps Shift+Symbol Shift+Z](#) and then [Caps Shift+Symbol Shift+0](#) on ZXUnGo+): Soft Reset
- [Ctrl+Alt+Backspace](#) ([Caps Shift+Symbol Shift+F](#) and then [B](#) on ZXUnGo+): 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 its contents using the command:

```
*CAT
```

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

```
CHAIN"MENU"
```

To load using the audio input:

```
*TAPE
CHAIN""
```

And then start playing the external audio device.

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

```
*DCAT
```

To put a particular virtual disc in a virtual drive:

```
*DIN discnum drivenum
```

Amstrad CPC 464

The [Amstrad CPC 464](#) was the first of a series of 8-bit home computers produced by Amstrad. The ZX-Uno version has been [made by McLeod](#).

Core features:

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



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

SD card format

This core does not use the SD card.

Keyboard

Special keys and buttons

The following ZXUnGo+ keyboard combinations are obtained while in **CPC** mode ([Caps Shift+Symbol Shift+F](#), then [U](#) and then [2](#)). Please read [the section dedicated to ZXUnGo+ keyboard modes](#) for more information.

During core execution:

- [Del](#) ([Caps Shift+Symbol Shift+V](#) on ZXUnGo+): **CLR**.
- [Print Scr](#) or [Left Windows](#) ([Caps Shift+Symbol Shift+T](#) on ZXUnGo+): **COPY**
- [Ctrl+Alt+Del](#) ([Caps Shift+Symbol Shift+F](#) and then [N](#) on ZXUnGo+): **Reset**.
- [Ctrl+Alt+Backspace](#) ([Caps Shift+Symbol Shift+F](#) and then [B](#) on ZXUnGo+): Hard reset. Backspace is the delete key, located in the top-right portion of the keyboard, above [Enter](#).
- [End](#) ([Caps Shift+Symbol Shift+B](#) on ZXUnGo+): 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.

ZX-Uno Amstrad CPC 6128 core is based on the [FPGAmstrad](#) project by Renaud Hélias.

Some of its features are:

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

SD card format

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

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

- [OS6128.ROM](#)
- [BASIC1-1.ROM](#)
- [AMSDOS.ROM](#)
- [MAXAM.ROM](#)

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

Copy all [ROM](#) and [DSK](#) files to the root directory of the FAT32 partition.

Keyboard

Special keys and buttons

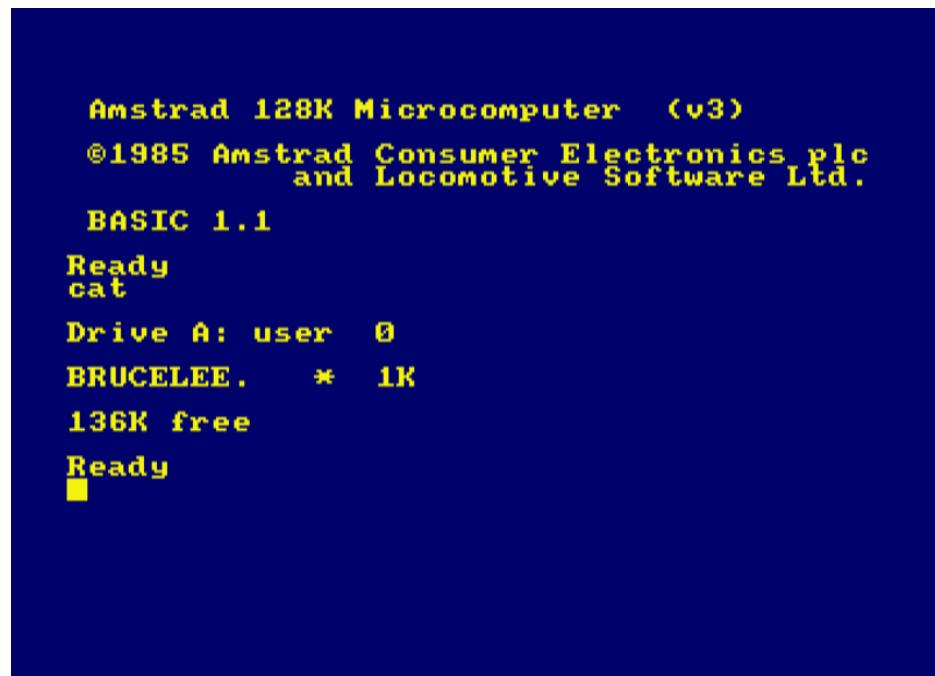
The following ZXUnGo+ keyboard combinations are obtained while in [CPC mode](#) ([Caps Shift+Symbol Shift+F](#), then [U](#) and then [2](#)). Please read [the section dedicated to ZXUnGo+ keyboard modes](#) for more information.

During core execution:

- [Page Up](#) ([Caps Shift+Symbol Shift+J](#) on ZXUnGo+): 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.

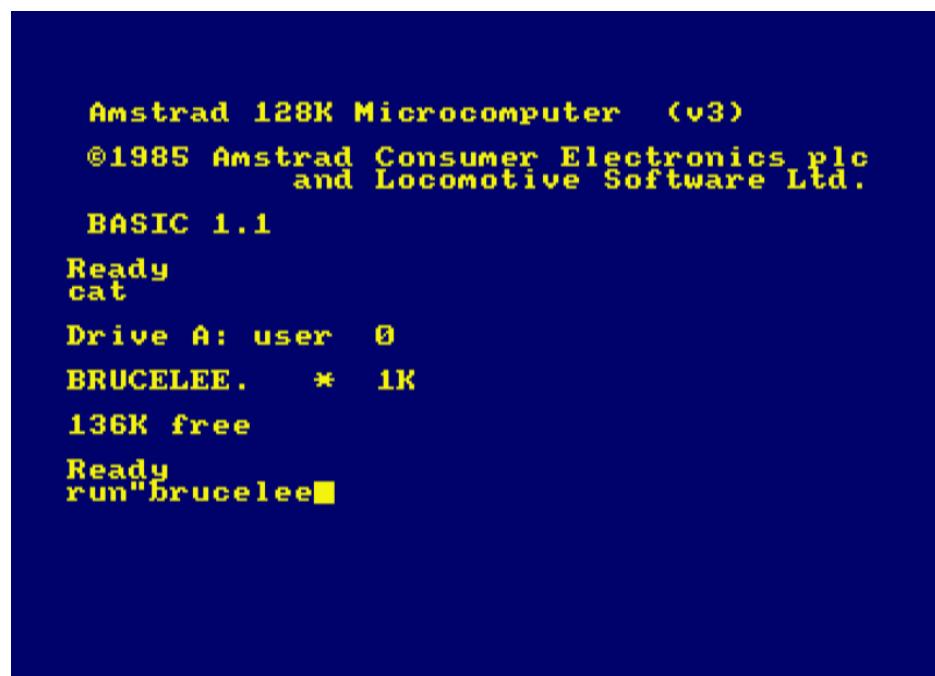
BASIC 1.1

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 I

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

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

Some of its features are:

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

SD card format

This core does not use the SD card.

Keyboard

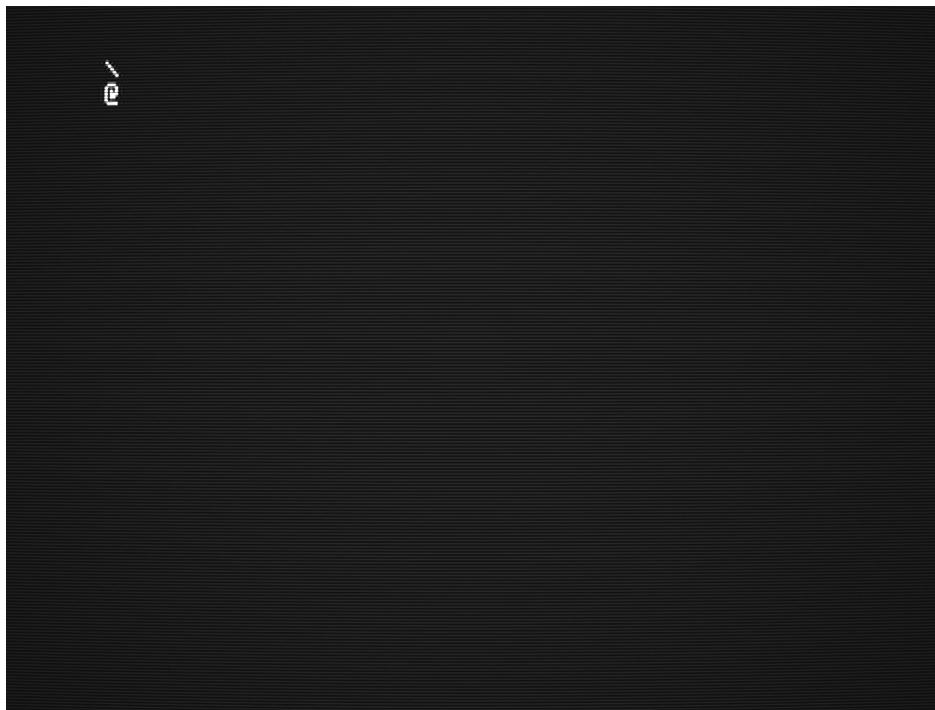
The keyboard uses an english layout.

The ZXUnGo+ keyboard combinations are obtained while in **APPLE** mode (**Caps Shift+Symbol Shift+F**, then **U** and then **8**). Please read [the section dedicated to ZXUnGo+ keyboard modes](#) for more information.

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

Basic Guide

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

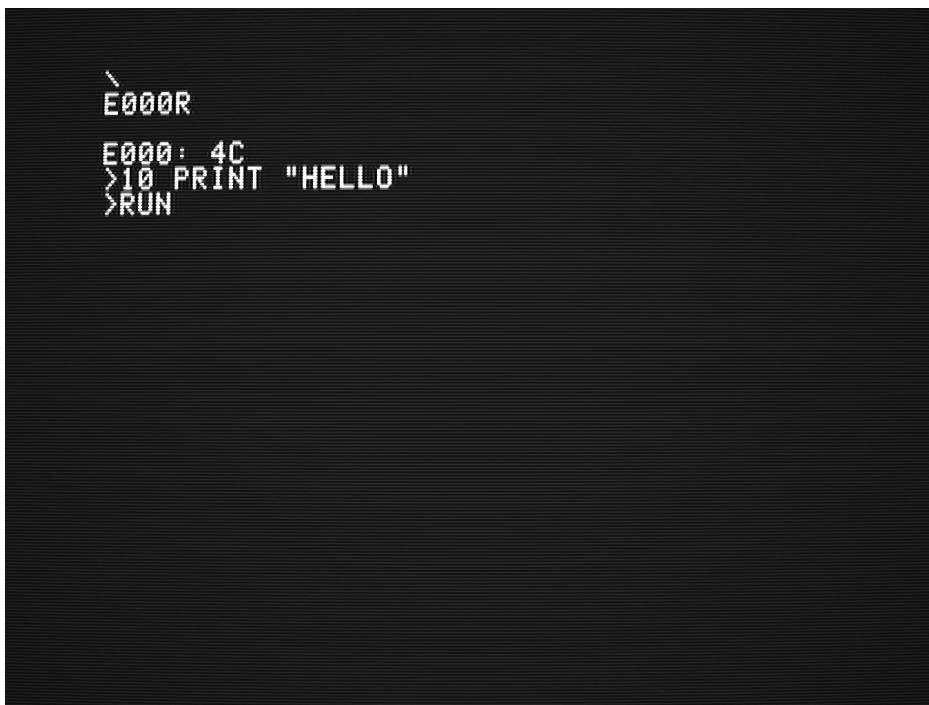


The BASIC interpreter included with this core is [Integer BASIC](#). To start it, you have to invoke the corresponding memory address (`0xE000`):

```
\  
E000R
```

Once the interpreter is loaded, the console changes in order to show that it is ready to accept [BASIC commands](#):

```
\  
E000R  
  
E000:4C  
>@
```



Apple II

Based on [Stephen A. Edwards Apple2fpga](#) and [vlait port for Papilio boards](#).

Some of its features are:

- Joystick Support (up to two fire 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 image.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



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

Special keys and buttons

The following ZXUnGo+ keyboard combinations are obtained while in **APPLE** mode (**Caps Shift+Symbol Shift+F**, then **U** and then **8**). Please read [the section dedicated to ZXUnGo+ keyboard modes](#) for more information.

While the core is running:

- - (numeric keyboard) (**Caps Shift+Symbol Shift+R** on ZXUnGo+): Enable or disable scanlines on VGA
- * (numeric keyboard) (**Caps Shift+Symbol Shift+E** on ZXUnGo+): Change between colour monitor and black and white
- **F1** (**Caps Shift+Symbol Shift+1** on ZXUnGo+) to **F10** (**Caps Shift+Symbol Shift+0** on ZXUnGo+): Insert disk image between 1 and 10 from the SD. Press **F12** afterwards.
- **Shift+F1** (**Caps Shift+Symbol Shift+A** and then **Caps Shift+Symbol Shift+1** on ZXUnGo+) to **Shift+F10** (**Caps Shift+Symbol Shift+A** and then **Caps Shift+Symbol Shift+0** on ZXUnGo+): Insert disk image between 11 and 20 from the SD. Press **F12** afterwards.
- **Ctrl+Alt+Backspace** (**Caps Shift+Symbol Shift+F** and then **B** on ZXUnGo+): Hard reset. Backspace is the delete key, located in the top-right portion of the keyboard, above **Enter**
- **F12** (**Caps Shift+Symbol Shift+W** on ZXUnGo+): Soft reset.

Arcades

Originally made for [Zx-Uuno Jamma Addon](#) (to connect inside an Arcade Machine), there are several [Arcade Game](#) cores. Afterwards, adapted versions compatible with joysticks (like ZXUnGo+ 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-Uuno 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

The following ZXUnGo+ keyboard combinations are obtained while in any keyboard mode except the normal [ZX](#) mode. Please read [the section dedicated to ZXUnGo+ keyboard modes](#) for more information.

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 ([Caps Shift+5](#), [Caps Shift+6](#), [Caps Shift+7](#) and [Caps Shift+8](#) on ZXUnGo+) 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
- [F10](#) ([Caps Shift+Symbol Shift+0](#) on ZXUnGo+): Soft Reset
- [Ctrl+Alt+Backspace](#) ([Caps Shift+Symbol Shift+F](#) and then [B](#) on ZXUnGo+): 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 ZX-Uno.

Keyboard

Special keys and buttons

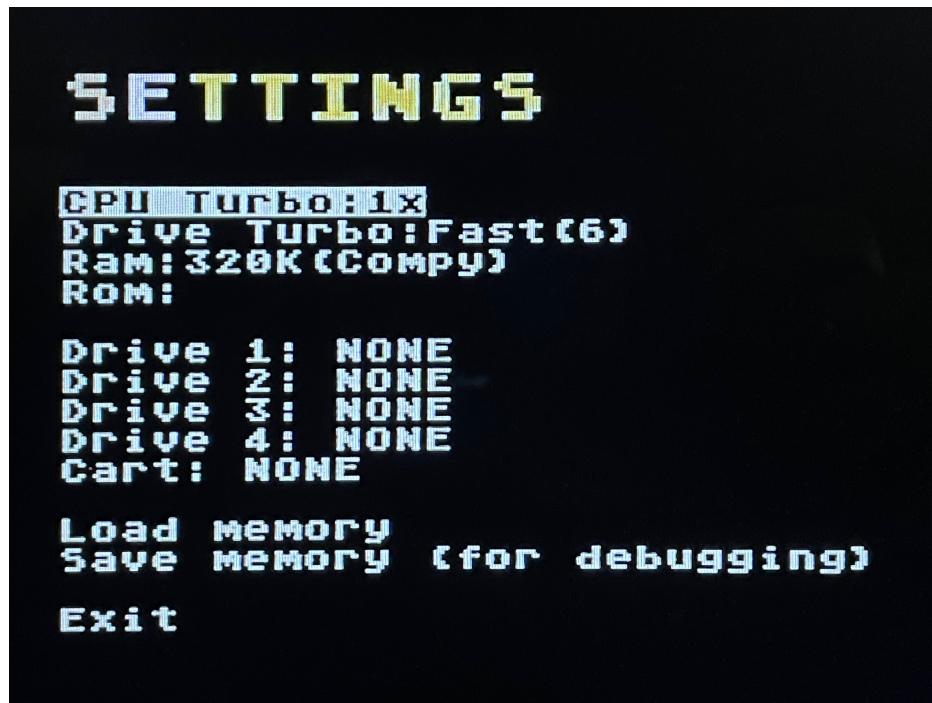
The following ZXUnGo+ keyboard combinations are obtained while in `ATARI` mode (`Caps Shift+Symbol Shift+F`, then `U` and then `5`). Please read [the section dedicated to ZXUnGo+ keyboard modes](#) for more information.

While the core is running:

- - (`Caps Shift+Symbol Shift+R` on ZXUnGo+): Enable or disable scanlines in VGA mode
- * (`Caps Shift+Symbol Shift+E` on ZXUnGo+): Change machine type between PAL and NTSC
- `Ctrl+Alt+Backspace` (`Caps Shift+Symbol Shift+F` and then `B` on ZXUnGo+): Hard reset.
- `F5` (`Caps Shift+Symbol Shift+5` on ZXUnGo+): `Help`
- `F6` (`Caps Shift+Symbol Shift+6` on ZXUnGo+): `Start`
- `F7` (`Caps Shift+Symbol Shift+7` on ZXUnGo+): `Select`
- `F9` (`Caps Shift+Symbol Shift+9` on ZXUnGo+): `Reset`
- `F11` (`Caps Shift+Symbol Shift+Q` on ZXUnGo+): Load a disk
- `F12` (`Caps Shift+Symbol Shift+W` on ZXUnGo+): 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** (**Caps Shift+Symbol Shift+W** on ZXUnGo+) shows or hides the configuration menu. Cursor keys (**Caps Shift+6** y **Caps Shift+7** on ZXUnGo+) 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).

ZX-Uno 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

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 ZX-Uno.

Keyboard

Special keys and buttons

The following ZXUnGo+ keyboard combinations are obtained while in **ATARI** mode (**Caps Shift+Symbol Shift+F**, then **U** and then **5**). Please read [the section dedicated to ZXUnGo+ keyboard modes](#) for more information.

During the core execution:

- **W, A, S, D** (customizable) or joystick 1: Directional controls for player 1
- **F** (customizable) or joystick 1 fire button: Player 1 fire button
- **I, J, K, L** (customizable) or joystick 2: Directional controls for player 2
- **H** (customizable) or joystick 2 fire button: Player 2 fire button
- **Ctrl+Alt+Backspace** (**Caps Shift+Symbol Shift+F** and then **B** on ZXUnGo+): Hard reset.

Control customization

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

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



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

Basic Guide

Pressing **Esc** (**Caps Shift+Space** on ZXUnGo+) or joystick button 2 shows or hides the configuration menu. Cursor keys (**Caps Shift+6** y **Caps Shift+7** on ZXUnGo+) and **Enter** to select and choose menu options.



The following options are available:

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

BBC Micro

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

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

Some of its features are:

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

SD card format

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

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

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

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



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

Keyboard



Special keys and buttons

The following ZXUnGo+ keyboard combinations are obtained while in **BBC** mode (**Caps Shift+Symbol Shift+F**, then **U** and then **6**). Please read [the section dedicated to ZXUnGo+ keyboard modes](#) for more information.

While the core is running:

- - (numeric keyboard) (**Caps Shift+Symbol Shift+R** on ZXUnGo+): Enable or disable scanlines in VGA mode
- F12 (**Caps Shift+Symbol Shift+W** on ZXUnGo+): Reset
- Shift+F12 (**Caps Shift+Symbol Shift+A** and then **Caps Shift+Symbol Shift+W** on ZXUnGo+): Soft Reset trying to load automatically the selected disk at **BEEB.MMB**
- Ctrl+Alt+Backspace (**Caps Shift+Symbol Shift+F** and then **B** on ZXUnGo+): Hard reset. Backspace is the delete key, located in the top-right portion of the keyboard, above **Enter**.

Basic guide



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

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

Or, for a core with Smart SPI

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

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

```
*CAT
```

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

```
*MENU
```

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

```
*DCAT
```

To load a particular virtual disc:

```
*DIN discnum
```



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

Computers Lynx

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

The ZX-Uno core has been [developed by Kyp069](#) and has these features:

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

SD card format

This core does not use the SD card

Keyboard

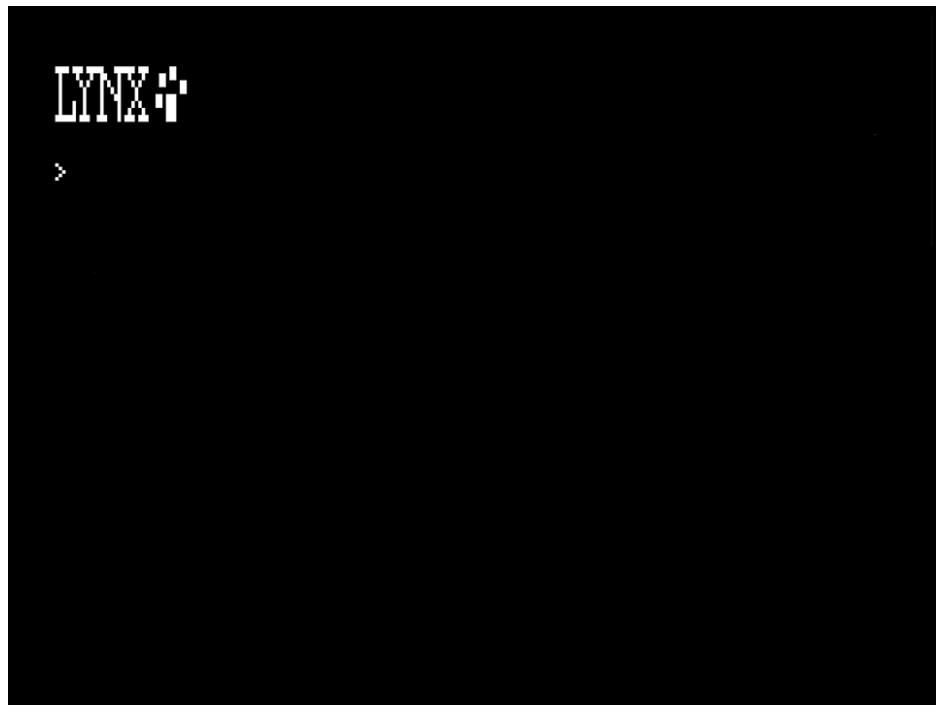
Special keys and buttons

The following ZXUnGo+ keyboard combinations are obtained while in [LYNX](#) mode ([Caps Shift+Symbol Shift+F](#), then [U](#) and then [M](#)). Please read [the section dedicated to ZXUnGo+ keyboard modes](#) for more information.

While running the core:

- [F6 \(Caps Shift+Symbol Shift+6 on ZXUnGo+\)](#): Switch between 48kB mode and 96kB mode (default)
- [F7 \(Caps Shift+Symbol Shift+7 on ZXUnGo+\)](#): Enable or disable Scorpio ROM
- [F8 \(Caps Shift+Symbol Shift+8 on ZXUnGo+\)](#): Switch the option to consider port \$80 bits 2 and 3, so that Level 9 games are displayed properly.
- [F11 \(Caps Shift+Symbol Shift+Q on ZXUnGo+\)](#): Reset.
- [Ctrl+Alt+Backspace \(Caps Shift+Symbol Shift+F and then B on ZXUnGo+\)](#): Hard reset. Backspace is the delete key, located in the top-right portion of the keyboard, above [Enter](#).

Basic Guide



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

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

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

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

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

You can use programs like [Lynx2Wav](#) with Lynx **TAP** files. The resulting audio files can be embedded inside of TSX or TZX with tools like MakeTSX or RetroConverter.

The [lince](#) script makes all this process easier, creating directly Maxduino **TZX** compatible files from Lynx **TAP** files.

CHIP-8

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

The ZX-Uno core is based on an existing [FPGA implementation](#) of the SuperChip.

There are several sites like [CHIP-8 Archive](#) or [Matthew Mikolay's CHIP-8](#) where you can obtain software for these machines.

SD card format

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

Keyboard

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

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

Special keys and buttons



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

Special keys and buttons

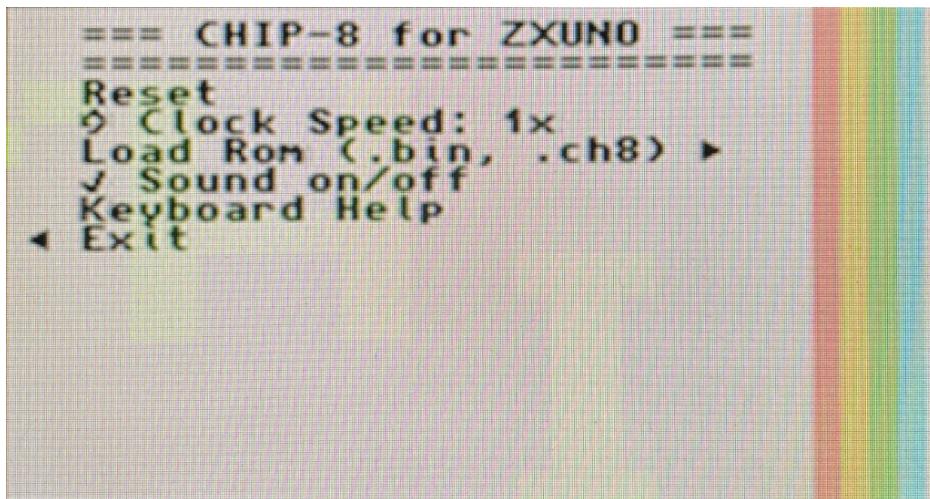
The following ZXUnGo+ keyboard combinations are obtained while in any keyboard mode except the normal **ZX** mode. Please read [the section dedicated to ZXUnGo+ keyboard modes](#) for more information.

While the core is running:

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

Basic Guide

Pressing **Esc** (**Caps Shift+Space** on ZXUnGo+) shows or hides the configuration menu. Use the cursor keys (**Caps Shift+6** and **Caps Shift+7** on ZXUnGo+), or **W** and **S**, and **Enter** to select and choose menu options.



The following options are available:

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

Colecovision

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

ZX-Uno 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 ZX-Uno.

Keyboard

Special keys and buttons

While the core is running:

- Cursor (**Caps Shift+5**, **Caps Shift+6**, **Caps Shift+7** and **Caps Shift+8** on ZXUnGo+) 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' (**Caps Shift+Space** on ZXUnGo+): 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' (**Caps Shift+Space** on ZXUnGo+) restarts the core and loads the ROM selection menu again.

Colour Genie

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

The ZX-Uno version has been [developed by Kyp069](#) and has these features:

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

SD card format

This core does not use the SD card.

Keyboard

The [Clear](#) key is not implemented.

Special keys and buttons

The following ZXUnGo+ keyboard combinations are obtained while in [eg2000](#) mode ([Caps Shift+Symbol Shift+F](#), then [U](#) and then [L](#)). Please read [the section dedicated to ZXUnGo+ keyboard modes](#) for more information.

While the core is running:

- Windows: MODSEL
- F5 ([Caps Shift+Symbol Shift+5](#) on ZXUnGo+): NMI
- F11 ([Caps Shift+Symbol Shift+Q](#) on ZXUnGo+) or Ctrl+Alt+Backspace ` ([Caps Shift+Symbol Shift+F](#) and then [B](#) on ZXUnGo+): Hard reset. Backspace is the delete key, located in the top-right portion of the keyboard, above [Enter](#)
- F12 ([Caps Shift+Symbol Shift+W](#) on ZXUnGo+) or [Ctrl+Alt+Del](#)([Caps Shift+Symbol Shift+F](#) and then [N](#) on ZXUnGo+): Reset

Basic Guide

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

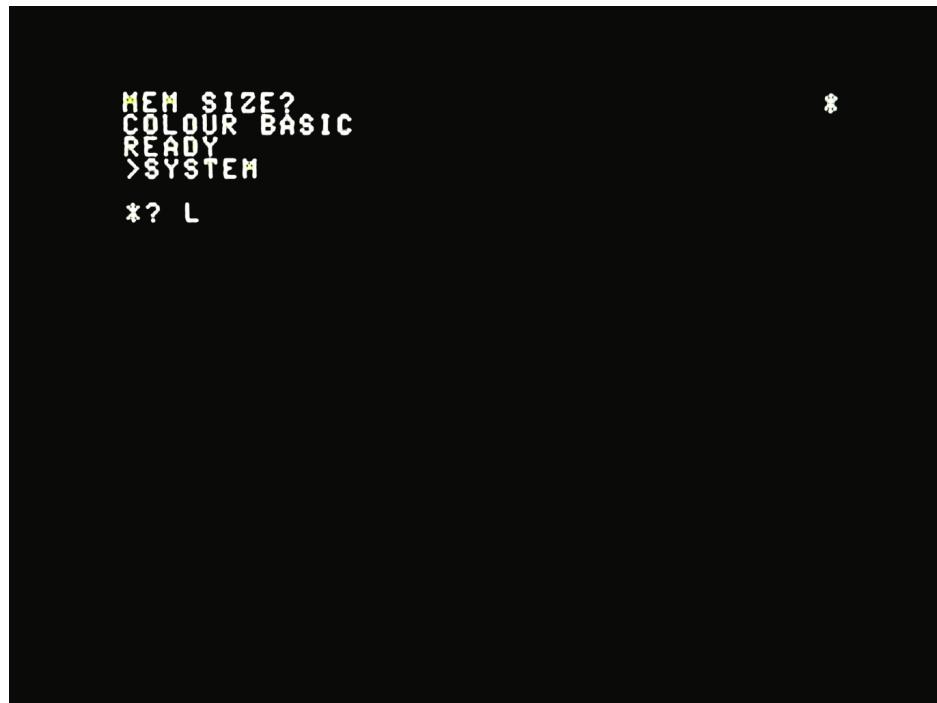


From BASIC, you can load software from a external tape or other external audio device with the following commands.

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

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

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



Commodore 16

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

The ZX-Uno core is based on [FPGATED project from István Hegedus](#), with some changes and upgrades, like loading tapes from audio sources.

Features:

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

SD card format

The SD card needs an exclusive format, which cannot be used with other cores. It's based on [D64](#) concatenated images, 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 image1.d64 dummyto256.bin image2.d64 dummyto256.bin (...) > c16rawsd.img
```

Dump the new file to the SD card, using [dd](#):

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

If it wasn't already, [install Commodore 16 core](#) into ZX-Uno.

Keyboard

Special keys and buttons

The following ZXUnGo+ keyboard combinations are obtained while in **C16 mode** (**Caps Shift+Symbol Shift+F**, then **U** and then **E**). Please read [the section dedicated to ZXUnGo+ keyboard modes](#) for more information.

While the core is running:

- **Esc** (**Caps Shift+Space** on ZXUnGo+): Esc
- **Tab**: RUN/STOP
- **Left Windows** (**Caps Shift+Symbol Shift+x** on ZXUnGo+): = Commodore
- Numeric Keyboard: Emulated Joystick
- **F1** (**Caps Shift+Symbol Shift+1** on ZXUnGo+) to **F3** (**Caps Shift+Symbol Shift+3** on ZXUnGo+): F1 to F3
- **Num Lock** (**Caps Shift+Symbol Shift+I** on ZXUnGo+) or **F4** (**Caps Shift+Symbol Shift+4** on ZXUnGo+): HELP
- **Insert** (**Caps Shift+Symbol Shift+G** on ZXUnGo+): Select the first disk in the SD card
- **Page Up** (**Caps Shift+Symbol Shift+J** on ZXUnGo+): Select the next disk
- **Page Down** (**Caps Shift+Symbol Shift+N** on ZXUnGo+): Select the previous disk
- **Ctrl+Page Up** (**Caps Shift+Symbol Shift+Z** and then **Caps Shift+Symbol Shift+J** on ZXUnGo+): Go 10 disks forward
- **Ctrl+Page Down** (**Caps Shift+Symbol Shift+Z** and then **Caps Shift+Symbol Shift+N** on ZXUnGo+): Go 10 disks back
- **+** (**Caps Shift+Symbol Shift+D** on ZXUnGo+): Change audio input polarity
- **-** (**Caps Shift+Symbol Shift+R** on ZXUnGo+): Enable or disable scanlines in VGA mode
- **Scroll Lock**: change between RGB and VGA video mode
- **F11** (**Caps Shift+Symbol Shift+Q** on ZXUnGo+): Change joystick between port 0 and port 1
- **F12** (**Caps Shift+Symbol Shift+W** on ZXUnGo+): Soft Reset
- **Ctrl+Alt+Backspace** (**Caps Shift+Symbol Shift+F** and then **B** on ZXUnGo+): Hard reset. Backspace is the delete key, located in the top-right portion of the keyboard, above **Enter**

Basic Guide



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



You can use programs like [Audiotap](#) or [Comodoro](#) with Commodore **TAP** files.

To show the contents of the current disk, press **F3** (**Caps Shift+Symbol Shift+3** on ZXUnGo+) 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 ZX-Uno core has been made by Quest.

Features:

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

SD card format

The SD card needs an exclusive format, which cannot be used with other cores. It's based on [D64](#) concatenated images, 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 [FB64.PRG](#) as the first program.



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



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

Windows

Concatenate the disk images using [COPY](#):

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

Dump the new file to the SD card, for example, using [HDD Raw Copy Tool](#).

macOS and Linux

Concatenate the disk images using [cat](#):

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

Dump the new file to the SD card, using [dd](#):

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

If it wasn't already, [install Commodore 64 core](#) into ZX-Uno.

Keyboard

Special keys and buttons

The following ZXUnGo+ keyboard combinations are obtained while in C64 mode (**Caps Shift+Symbol Shift+F**, then **U** and then **4**). Please read [the section dedicated to ZXUnGo+ keyboard modes](#) for more information.

While the core is running:

- **Esc** (**Caps Shift+Space** on ZXUnGo+): Start/Stop
- **Tab**: RUN/STOP
- **Alt** (**Caps Shift+Symbol Shift+C** on ZXUnGo+): = Commodore
- Numeric Keyboard: Emulated Joystick
- **F1** (**Caps Shift+Symbol Shift+1** on ZXUnGo+) to **F8** (**Caps Shift+Symbol Shift+8** on ZXUnGo+): F1 to F8
- **F9** (**Caps Shift+Symbol Shift+9** on ZXUnGo+): Pound symbol key
- **F10** (**Caps Shift+Symbol Shift+0** on ZXUnGo+): + symbol key
- **Insert** (**Caps Shift+Symbol Shift+G** on ZXUnGo+): Select the first disk in the SD card
- **Page Up** (**Caps Shift+Symbol Shift+J** on ZXUnGo+): Select the next disk
- **Page Down** (**Caps Shift+Symbol Shift+N** on ZXUnGo+): Select the previous disk
- **Ctrl+Page Up** (**Caps Shift+Symbol Shift+Z** and then **Caps Shift+Symbol Shift+J** on ZXUnGo+): Go 10 disks forward
- **Ctrl+Page Down** (**Caps Shift+Symbol Shift+Z** and then **Caps Shift+Symbol Shift+N** on ZXUnGo+): Go 10 disks back
- **+** (**Caps Shift+Symbol Shift+D** on ZXUnGo+): Change audio input polarity
- **-** (**Caps Shift+Symbol Shift+R** on ZXUnGo+): Enable or disable scanlines in VGA mode
- **End** (**Caps Shift+Symbol Shift+B** on ZXUnGo+): Change the colour palette
- **F11** (**Caps Shift+Symbol Shift+Q** on ZXUnGo+): Change joystick port behaviour
- **Ctrl+F12** (**Caps Shift+Symbol Shift+W** on ZXUnGo+): Soft Reset to JyffyDOS mode
- **Ctrl+F12** (**Caps Shift+Symbol Shift+Z** and then **Caps Shift+Symbol Shift+W** on ZXUnGo+): Soft Reset to original C64 ROM mode (to load from tape)
- **Ctrl+Alt+Backspace** (**Caps Shift+Symbol Shift+F** and then **B** on ZXUnGo+): Hard reset. Backspace is the delete key, located in the top-right portion of the keyboard, above **Enter**

Basic Guide



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



To be able to load from audio tape, you have to use the original C64 ROM, which is enabled using **Ctrl+F12** (**Caps Shift+Symbol Shift+Z** and then **Caps Shift+Symbol Shift+W** on ZXUnGo+).



You can use programs like [Audiotap](#) or [Comodoro](#) with Commodore **TAP** files.

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

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

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



To be able to load from disk, you have to use the JiffyDOS C64 ROM, which is enabled using **F12** (**Caps Shift+Symbol Shift+W** on ZXUnGo+).

Commodore PET

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

The ZX-Uno version has been [made by Jepalza](#), based on the [pet2001fpga](#) project.

Features:

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

SD card format

This core does not use the SD card

Keyboard

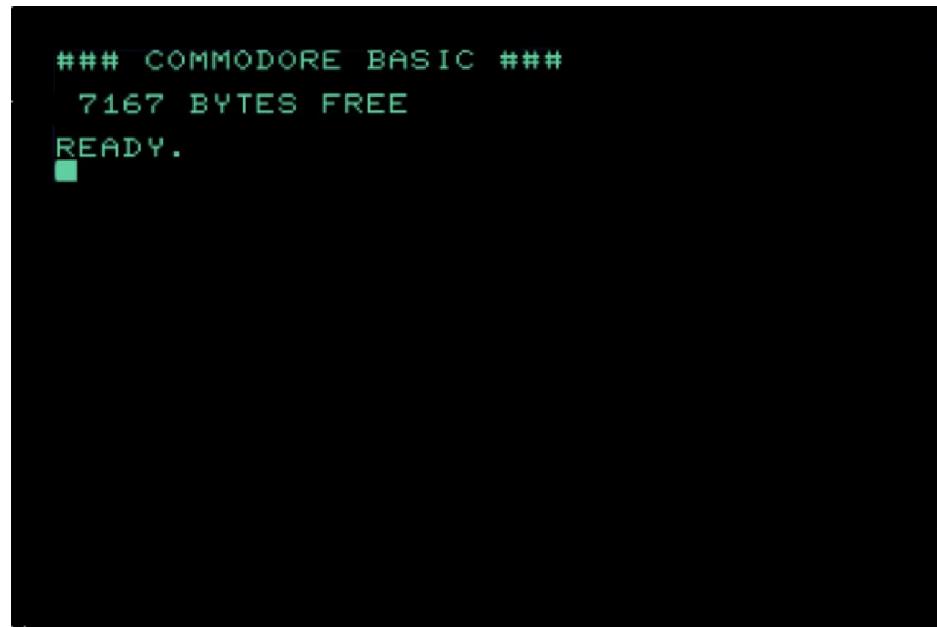
Special keys and buttons

The following ZXUnGo+ keyboard combinations are obtained while in **CPET** mode (**Caps Shift+Symbol Shift+F**, then **U** and then **K**). Please read [the section dedicated to ZXUnGo+ keyboard modes](#) for more information.

While running the core:

- **F1** (**Caps Shift+Symbol Shift+1** on ZXUnGo+): Cancel tape loading
- **Alt** (**Caps Shift+Symbol Shift+C** on ZXUnGo+): Type using graphic mode

Basic Guide



```
### COMMODORE BASIC ###
7167 BYTES FREE
READY.
```

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.



You can use programs like [Audiotap](#) or [Comodore](#) with Commodore **TAP** files.

Commodore VIC-20

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

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

Main features:

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

SD card format

This core does not use the SD card

Keyboard

Special keys and buttons

The following ZXUnGo+ keyboard combinations are obtained while in **VIC** mode ([Caps Shift+Symbol Shift+F](#), then [U](#) and then [9](#)). Please read [the section dedicated to ZXUnGo+ keyboard modes](#) for more information.

While the core is running:

- **F9** ([Caps Shift+Symbol Shift+9](#) on ZXUnGo+) to **F11** ([Caps Shift+Symbol Shift+Q](#) on ZXUnGo+): Load cartridges embedded in the core
- Numeric keyboard - ([Caps Shift+Symbol Shift+R](#) on ZXUnGo+): Change audio input polarity. The red LED is enable when inverted.
- **Num Lock** ([Caps Shift+Symbol Shift+I](#) on ZXUnGo+): Reset disabling the 32K memory expansion (enabled by default)
- **F12** ([Caps Shift+Symbol Shift+W](#) on ZXUnGo+): Reset enabling the memory expansion
- **Ctrl+Alt+Backspace** ([Caps Shift+Symbol Shift+F](#) and then [B](#) on ZXUnGo+): Hard reset. Backspace is the delete key, located in the top-right portion of the keyboard, above [Enter](#)

Basic Guide



```
*** CBM BASIC V2 ***
28159 BYTES FREE
READY.
```

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.



You can use programs like [Audiotap](#) or [Comodoro](#) with Commodore **TAP** files.

Flappy Bird

[Flappy Bird](#) was very successful mobile game. The main goal is to control a bird, attempting to fly between columns of green pipes without hitting them.

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

Main features:

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

SD card format

This core does not use the SD card

Keyboard

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

Special keys and buttons

While the core is running:

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

Galaksija

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

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

ZX-Uno core features:

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

SD card format

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

Keyboard

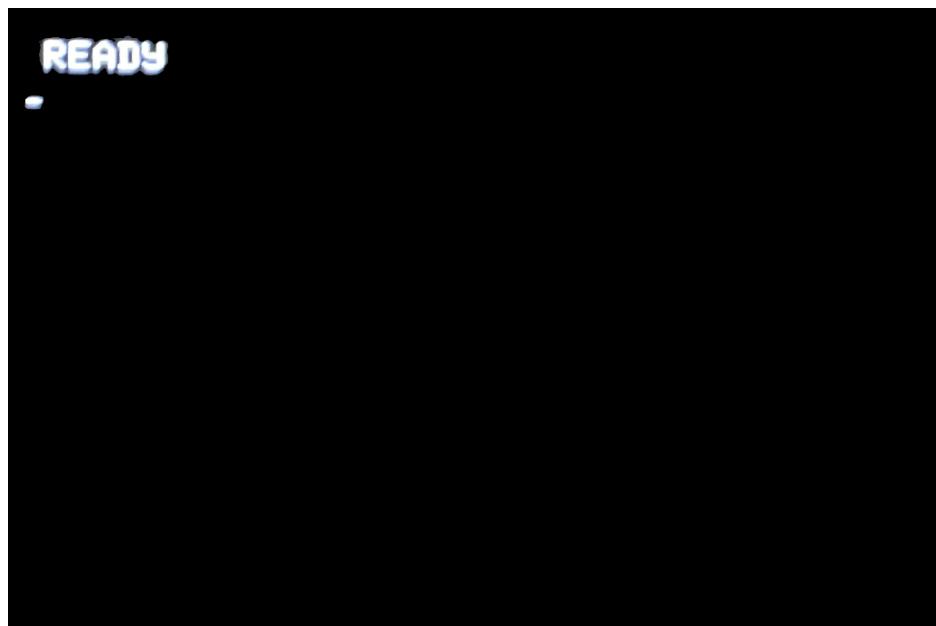
The following ZXUnGo+ keyboard combinations are obtained while in **GALAKSIJ** mode (**Caps Shift+Symbol Shift+F**, then **U** and then **G**). Please read [the section dedicated to ZXUnGo+ keyboard modes](#) for more information.

Special keys and buttons

While the core is running:

- **F1** (**Caps Shift+Symbol Shift+1** on ZXUnGo+): Normal speed
- **F2** (**Caps Shift+Symbol Shift+2** on ZXUnGo+): 2x speed
- **F3** (**Caps Shift+Symbol Shift+3** on ZXUnGo+): 4x speed
- **F9** (**Caps Shift+Symbol Shift+9** on ZXUnGo+): Enable or disable sound
- **Esc** (**Caps Shift+Space** on ZXUnGo+): Freezes a BASIC or machine code execution. Pressing again the same key, resumes the execution
- **Del** (**Caps Shift+Symbol Shift+V** on ZXUnGo+): Pauses a BASIC program execution while being pressed
- **End** (**Caps Shift+Symbol Shift+B** on ZXUnGo+): BREAKs a BASIC program execution
- **Shift.+M**: Enables plus mode in those programs that have the support (e.g. TETRIS.GTP)
- **M**: Gets back to normal mode from plus mode
- **F12** (**Caps Shift+Symbol Shift+W** on ZXUnGo+): Soft Reset
- **Ctrl+Alt+F5**: NMI
- **Ctrl+Alt+Del** (**Caps Shift+Symbol Shift+F** and then **N** on ZXUnGo+): Soft Reset
- **Ctrl+Alt+Backspace** (**Caps Shift+Symbol Shift+F** and then **B** on ZXUnGo+): Hard reset. Backspace is the delete key, located in the top-right portion of the keyboard, above **Enter**

Basic guide



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

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

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

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

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

You can check the available BASIC commands [at Wikipedia](#).

HT-1080Z

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

The core is based [Jozsef Laszlo's original work](#) for MiST, ported by azesmbog to the AEON development board, and now to ZX-Uno.

It's main features are:

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

SD card format

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

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

You can obtain file conversion utilities at this sites:

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

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

<https://github.com/pullmoll/cass80>

Keyboard

Special keys and buttons

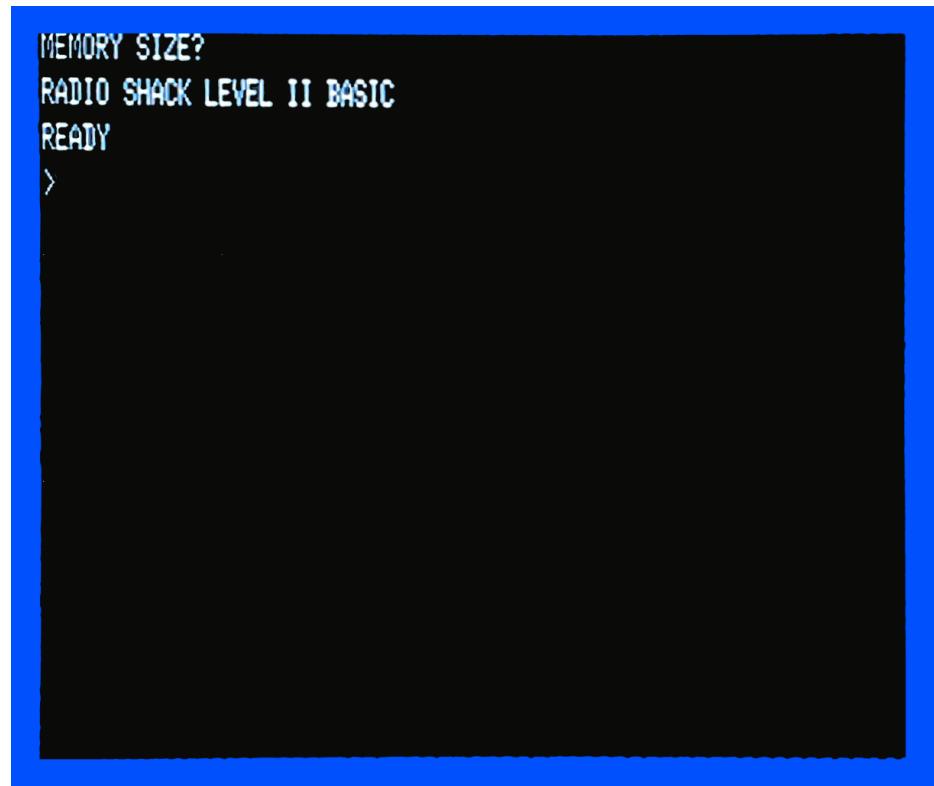
The following ZXUnGo+ keyboard combinations are obtained while in **TRS80** mode (**Caps Shift+Symbol Shift+F**, then **U** and then **F**). Please read [the section dedicated to ZXUnGo+ keyboard modes](#) for more information.

While the core is running:

- **F5** (**Caps Shift+Symbol Shift+5** on ZXUnGo+): Change ink colour
- **F6** (**Caps Shift+Symbol Shift+6** on ZXUnGo+): Change paper colour
- **F7** (**Caps Shift+Symbol Shift+7** on ZXUnGo+): Change border colour
- **F9** (**Caps Shift+Symbol Shift+9** on ZXUnGo+): Select 64 or 32 columns mode (only with VGA output)
- **F10** (**Caps Shift+Symbol Shift+0** on ZXUnGo+): **PAGE** key
- **Home** (**Caps Shift+Symbol Shift+H** on ZXUnGo+): **CLEAR** key
- **Shift+0** (**Caps Shift+Symbol Shift+A** and then **0** on ZXUnGo+): If **ULCBAS.CAS** has been loaded (Upper/Lower Case Driver) changes between only uppercase mode or uppercase and lowercase mode
- **Ctrl+Alt+Del** (**Caps Shift+Symbol Shift+F** and then **N** on ZXUnGo+): Soft Reset
- **Ctrl+Alt+Backspace** (**Caps Shift+Symbol Shift+F** and then **B** on ZXUnGo+): Hard reset. Backspace is the delete key, located in the top-right portion of the keyboard, above **Enter**

Basic Guide

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



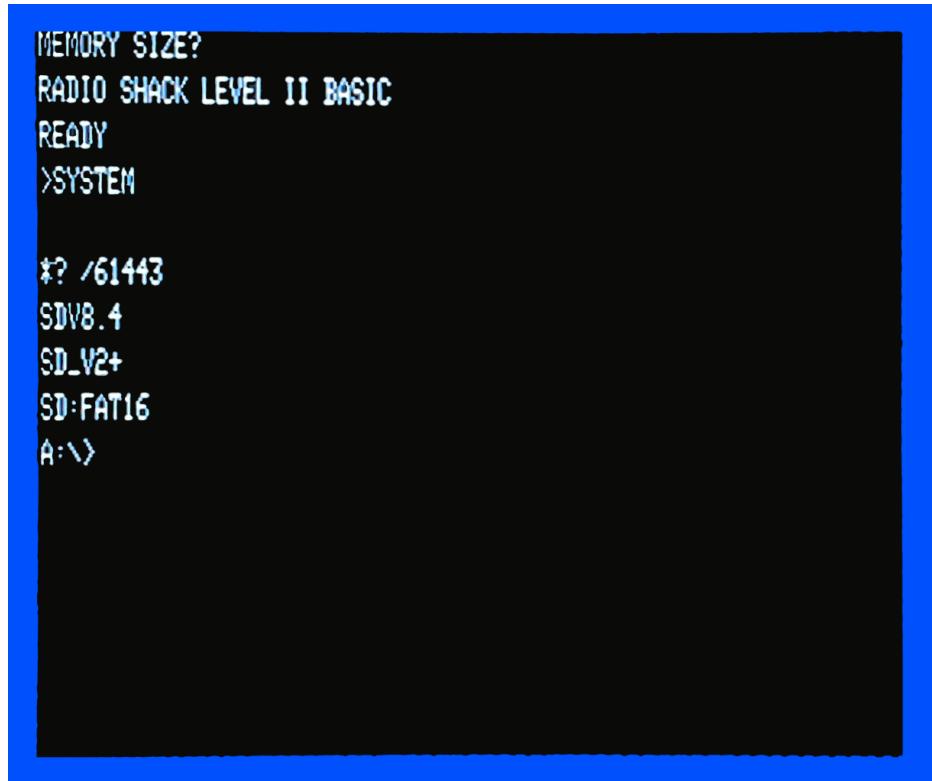
After the BASIC interpreter has loaded, if, for example, you type **PRINT MEM**, the available memory for BASIC will be displayed.

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

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

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

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



The screenshot shows the SDOS interface with the following text displayed:

```
MEMORY SIZE?  
RADIO SHACK LEVEL II BASIC  
READY  
>SYSTEM  
  
*? /61443  
SDV8.4  
SD_V2+  
SD:FAT16  
A:\>
```

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



All SDOS commands have to be typed in uppercase.



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



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

Jupiter ACE

The [Jupiter ACE](#) was a british micro computer produced by Jupiter Cantab in the early 1980s.

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

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

SD card format

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

Keyboard

Special keys and buttons

The following ZXUnGo+ keyboard combinations are obtained while in **ACE mode** ([Caps Shift+Symbol Shift+F](#), then [U](#) and then [C](#)). Please read [the section dedicated to ZXUnGo+ keyboard modes](#) for more information.

While the core is running:

- [F1 \(Caps Shift+Symbol Shift+1](#) on ZXUnGo+) to [F4 \(Caps Shift+Symbol Shift+4](#) on ZXUnGo+): Turbo Modes
- [F12 \(Caps Shift+Symbol Shift+W](#) on ZXUnGo+): Soft Reset
- [Ctrl+Alt+F5 \(Caps Shift+Symbol Shift+Y](#) on ZXUnGo+): NMI
- [Ctrl+Alt+Del \(Caps Shift+Symbol Shift+F](#) and then [N](#) on ZXUnGo+): Reset
- [Ctrl+Alt+Backspace \(Caps Shift+Symbol Shift+F](#) and then [B](#) on ZXUnGo+): Hard reset. Backspace is the delete key, located in the top-right portion of the keyboard, above [Enter](#)

Basic Guide

From within Forth, you can load from a external tape or other external audio device with the command **LOAD <nombre>**.

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

```
61440 call
```

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



All SDOS commands have to be typed in uppercase.

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



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

```
vlist
FORTH UFLOAT INT FNEGATE F\ F+ F-
+F- LOAD BUVERIFY VERIFY BLOAD B
SAVE SAVE LIST EDIT FORGET REDEF
INE EXIT ." ( )+LOOP LOOP DO UN
TIL REPEAT BEGIN THEN ELSE WHILE
WHILE J LEAVE J I DEFINITIONS V
OCABULARY IMMEDIATE RUNS> DOES>
COMPILER CALL DEFINER ASCII LITE
RAL CONSTANT VARIABLE ALLOT C,
CREATE : DECIMAL MIN MAX XOR AN
D DR S- 1- S+ 1+ D+ - + DNEGATE
NEGATE U/MOD */ * MOD / */MOD /M
OD U* D< U< <> = @> @< @= ABS O
UT IN INKEY BEEP PLOT AT F. EMIT
CR SPACES SPACE HOLD CLS # ##$ U
^ SIGN #> <# TYPE ROLL PICK DU
BR ROT ?DUP R> >R ! @ C! C@ SWAP
DROP DUP SLOW FAST INVIS VIS CO
NUERT NUMBER EXECUTE FIND VLIST
WORD RETYPE QUERY LINE ; PAD BAS
E CURRENT CONTEXT HERE ABORT QUI
T OK
■
```

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 ZX-Uno.

Keyboard

Special keys and buttons

The following ZXUnGo+ keyboard combinations are obtained while in **MSX** mode (**Caps Shift+Symbol Shift+F**, then **U** and then **3**). Please read [the section dedicated to ZXUnGo+ keyboard modes](#) for more information.

While running the core:

- **F1** (**Caps Shift+Symbol Shift+1** on ZXUnGo+) to **F5** (**Caps Shift+Symbol Shift+5** on ZXUnGo+): **F1** to **F5**
- **Print Scr** (**Caps Shift+Symbol Shift+T** on ZXUnGo+): Changes between VGA and RGB mode
- **Scroll Lock** (**Caps Shift+Symbol Shift+Y** on ZXUnGo+): Enables or disables scanlines
- **Pause** (**Caps Shift+Symbol Shift+U** on ZXUnGo+): Changes between 50Hz and 60Hz
- **F11** (**Caps Shift+Symbol Shift+Q** on ZXUnGo+): Enables and disables turbo mode
- **Ctrl+Alt+Del** (**Caps Shift+Symbol Shift+F** and then **N** on ZXUnGo+): Soft Reset
- **Ctrl+Alt+Del** (**Caps Shift+Symbol Shift+F** and then **N** on ZXUnGo+): Reset
- **Ctrl+Alt+Backspace** (**Caps Shift+Symbol Shift+F** and then **B** on ZXUnGo+): Hard reset. Backspace is the delete key, located in the top-right portion of the keyboard, above **Enter**
- **Left ALT** (**Caps Shift+Symbol Shift+C** on ZXUnGo+): MSX GRAPH
- **Right ALT** (**Caps Shift+Symbol Shift+M** on ZXUnGo+): MSX CODE
- **Page Up** (**Caps Shift+Symbol Shift+J** on ZXUnGo+): MSX SELECT
- **Home** (**Caps Shift+Symbol Shift+H** on ZXUnGo+): MSX HOME (**Shift+HOME** or **Caps Shift+Symbol Shift+A** and then **Caps Shift+Symbol Shift+H** on ZXUnGo+): CLS)
- **End** (**Caps Shift+Symbol Shift+B** on ZXUnGo+): MSX STOP
- **Ñ or Windows** (`**Caps Shift+Symbol Shift+X** on ZXUnGo+`): MSX DEAD



In BASIC use **CTRL+STOP** (**Ctrl+End** or **Caps Shift+Symbol Shift+Z** and then **Caps Shift+Symbol Shift+B** on ZXUnGo+) 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.



MSX BASIC version 1.0
Copyright 1983 by Microsoft
28815 Bytes free
Ok
■

color auto goto list run

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 ZX-Uno 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
- The basic core uses all 512 KB ZX-Uno 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.
- There are another core versions adapted to use 2MB of SRAM and which can load games using up to 1MB of CHR RAM and 1MB of PRG RAM (2MB in total)
- 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 ZX-Uno.

Keyboard

Special keys and buttons

The following ZXUnGo+ keyboard combinations are obtained while in any keyboard mode except the normal [ZX](#) mode. Please read [the section dedicated to ZXUnGo+ keyboard modes](#) for more information.

While the core is running:

- [Esc](#) ([Caps Shift+Space](#) on ZXUnGo+) or joystick button 2: Show or hide configuration menu
- Cursor keys ([Caps Shift+6](#) and [Caps Shift+7](#) on ZXUnGo+), and [Enter](#) to use the menu
- [Ctrl+Alt+Backspace](#) ([Caps Shift+Symbol Shift+F](#) and then [B](#) on ZXUnGo+): Hard reset
- [W, A, S, D](#) (customizable) or joystick 1: Directional controls for player 1

- **F** and **R** (customizable) or joystick 1 fire buttons: Player 1 A and B buttons
- **5 y 1:** **Select** and **Start** (Player 1)
- **I, J, K, L** (customizable) or joystick 2: Directional controls for player 2
- **H, Y** (customizable) or joystick 2 fire buttons: Player 2 A and B button
- **6 y 2:** **Select** and **Start** (Player 1)
- **F4** (**Caps Shift+Symbol Shift+F4** on ZXUnGo+): Swap the keyboard keys for player 1 and player 2
- **Ctrl+Alt+Backspace** (**Caps Shift+Symbol Shift+F** and then **B** on ZXUnGo+): Hard reset. Backspace is the delete key, located in the top-right portion of the keyboard, above **Enter**

Control customization

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

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



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

Basic Guide

Pressing **Esc** (**Caps Shift+Space** on ZXUnGo+) or joystick button 2 shows or hides the configuration menu. To navigate the menu and activate or choose any option, use the cursor keys (**Caps Shift+6** and **Caps Shift+7** on ZXUnGo+) and **Enter**.



The following options are available:

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



You can move faster through the menu options using **Page Up** (**Caps Shift+Symbol Shift+J** on ZXUnGo+) and **Page Down** (**Caps Shift+Symbol Shift+N** on ZXUnGo+).

Ondra SPO 186

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

The [ZX-Uno version](#) has been developed by PetrM1, based on the previous MiSTER version, and has these features:

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

SD card format

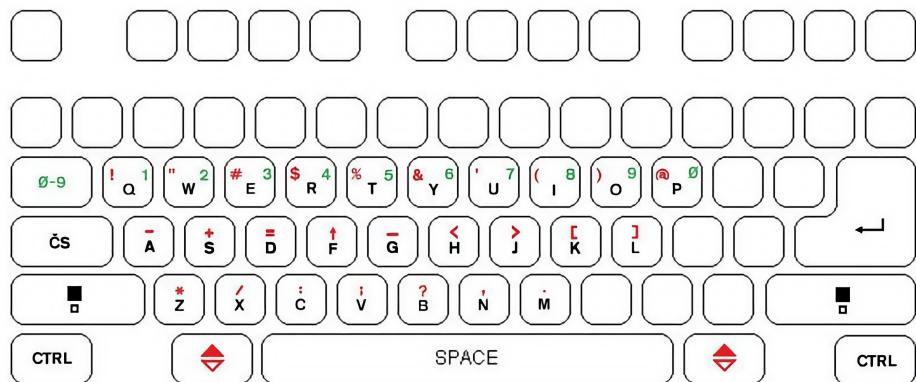
This core does not use the SD card.

Keyboard



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

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



Special keys and buttons

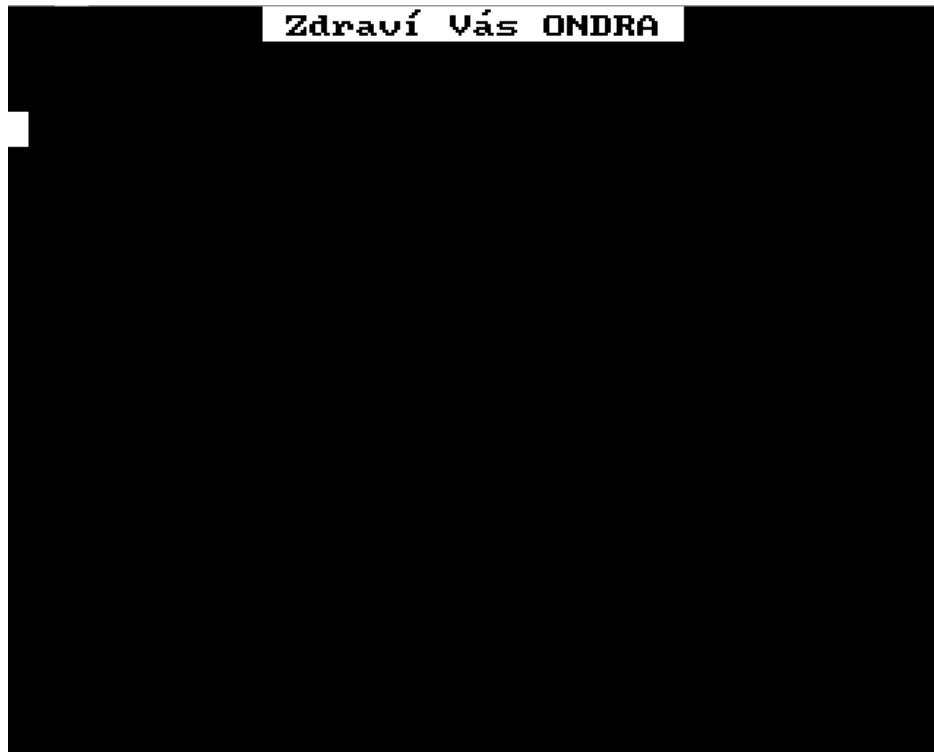
- **Shift:** Shift
- **Alt:** Symbols
- **Tab:** Numbers
- **Control:** Ctrl
- **Shift:** ČS (used for Czech diacritics)

Basic guide



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

.KÓD 1



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

Oric Atmos

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

It has been [developed by Kyp069](#) and has these features:

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

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

SD card format

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



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

Windows

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

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

Dump the new file to the SD card, for example, using [HDD Raw Copy Tool](#).

macOS and Linux

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

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

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

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

Keyboard

This is the keyboard layout:



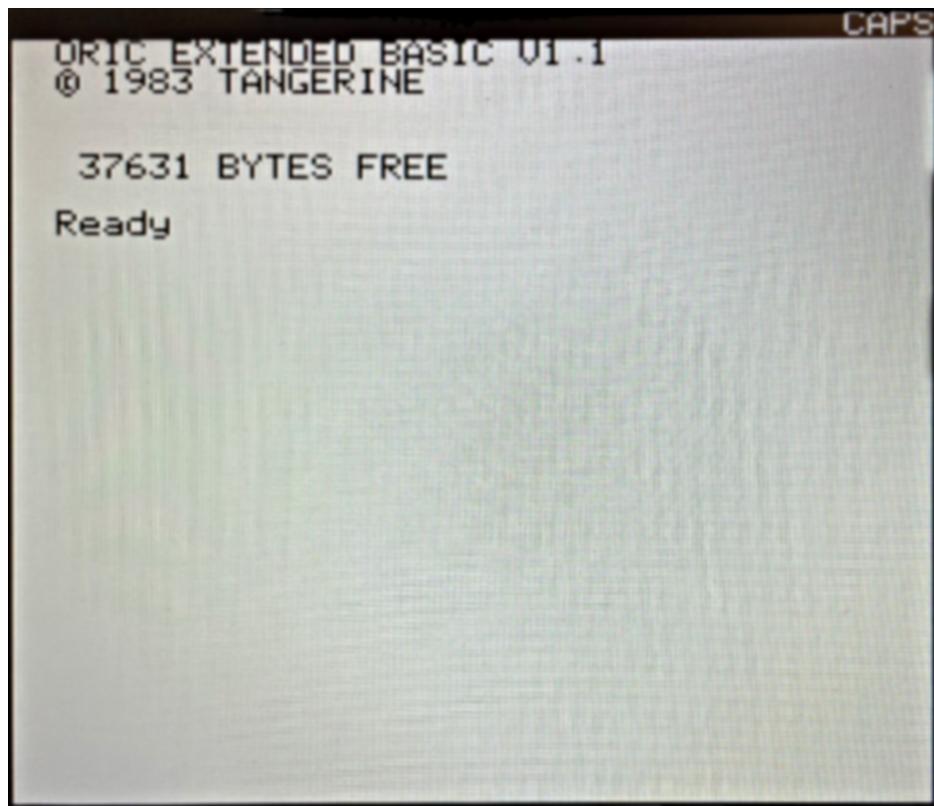
Special keys and buttons

The following ZXUnGo+ keyboard combinations are obtained while in ORIC mode (**Caps Shift+Symbol** **Shift+F**, then **U** and then **A**). Please read [the section dedicated to ZXUnGo+ keyboard modes](#) for more information.

While the core is running:

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

Basic Guide



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

```
CALL800
```

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

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

```
-DEFENDER.EXE
```

Oric Atmos (Kyp)

A different core version of [Oric Atmos](#), based on Rampa's version for MiST, Mistica and SiDi.

This version has the following features:

- Composite or VGA (Scandoubler) video out
- 64KB RAM and 16KB ROM. The ROM hides the upper memory, but all the memory banks are accessible using machine code
- Selectable ROM between the original version (1.1), and versión 1.22, alternate ROM which fixes errors
- The joystick directional controls are mapped to cursor keys ([Caps Shift+5](#), [Caps Shift+6](#), [Caps Shift+7](#) and [Caps Shift+8](#) on ZXUnGo+), and both fire buttons to the keys [Space](#) and [S](#).
- Program loading through the audio input
- Program saving using the audio output

For more information you can see [RetroWiki forums](#).

SD card format

This core does not use the SD card.

Keyboard

The keyboard distribution of a PS/2 PC keyboard tries to be as similar as possible to a real Oric Atmos keyboard.



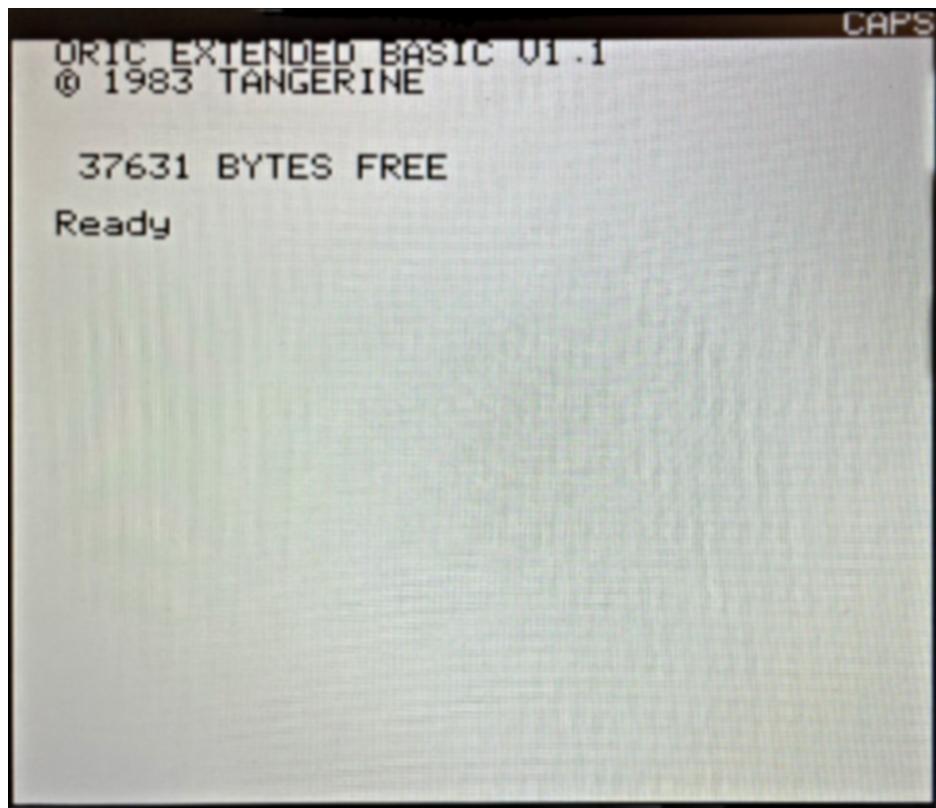
Special keys and buttons

The following ZXUnGo+ keyboard combinations are obtained while in ORIC mode (**Caps Shift+Symbol Shift+F**, then **U** and then **A**). Please read [the section dedicated to ZXUnGo+ keyboard modes](#) for more information.

While the core is running:

- **F1** (**Caps Shift+Symbol Shift+1** on ZXUnGo+): Enable ROM 1.1
- **F2** (**Caps Shift+Symbol Shift+2** on ZXUnGo+): Enable ROM 1.22
- **F5** (**Caps Shift+Symbol Shift+5** on ZXUnGo+): NMI
- **F10** (**Caps Shift+Symbol Shift+0** on ZXUnGo+): Enable audio out when using **CSAVE**
- **Ctrl+Alt+Backspace** (**Caps Shift+Symbol Shift+F** and then **B** on ZXUnGo+) or **F11** (**Caps Shift+Symbol Shift+Q** on ZXUnGo+): Hard reset. Backspace is the delete key, located in the top-right portion of the keyboard, above **Enter**
- **Ctrl+Alt+Del** (**Caps Shift+Symbol Shift+F** and then **N** on ZXUnGo+) or **F12** (**Caps Shift+Symbol Shift+W** on ZXUnGo+): Soft reset
- **Scroll Lock** (**Caps Shift+Symbol Shift+Y** on ZXUnGo+): to change between composite and VGA video
- **Alt Gr.** (**Caps Shift+Symbol Shift+M** on ZXUnGo+): **FUNC** key
- **Esc** (**Caps Shift+Space** on ZXUnGo+): **ESC** key
- **Del** (**Caps Shift+Symbol Shift+V** on ZXUnGo+): **DEL** key
- **Ctrl** (**Caps Shift+Symbol Shift+Z** on ZXUnGo+): **CTRL** key
- **Shift** (**Caps Shift** on ZXUnGo+): **SHIFT** key
- **Windows Left, Alt Left, Windows Right, Menu**: Cursor keys **Left, Down, Up, Right**

Basic Guide



From BASIC, you can load software from a external tape or other external audio device using **CLOAD** and **Enter** to load BASIC programs. Then, the following message will appear on screen

Searching...

When a program load is detected, the text changes to

Loading...<program name>

At [RetroWiki forums](#) there's a Linux script named `calorico.sh`, using bash and PHP, along with `TSXphpclass` utilities by [Natalia Pujol](#) which converts `.TAP` Oric files into `.TSX`.



To save a BASIC program with the audio output, you can use the command `CSAVE "NAME"` and `Enter`. If you want to hear while recording, enable the recording audio previously using the `F10` key (`Caps Shift+Symbol Shift+0` on ZXUnGo+).



Do not keep recording audio enabled (`F10` or `Caps Shift+Symbol Shift+Z` on ZXUnGo+) while loading using the audio input. This is because the audio feedback has a high pitch and volume sound which breaks audio loading.

PC XT

[Next186lite](#) core is an implementation of [Next186](#) core for FleaFPGA and Papilio Pro, 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 in FAT16 format and with MS-DOS (or similar) installed. You can achieve this using, for example, virtualization software and connecting directly de SD card device as a hard disk. You can also find several pre-made images at [ZX-Uno forum page](#) and [ZXDOS+ forums](#).

Once partitioned and formatted (with a reserve at the end, see later), you have to save the BIOS image [Next186_BIOS_zxdos_ddr3.COM](#) (available [at this link](#)) in the last 16 sectors of the SD card.



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



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

Windows

You can use [HxD \(Hex Editor for Windows\)](#):

1. Run the program with administrator privileges and select the option **Open disk...** from the **Tools** menu
2. Uncheck the option to only read and select the SD card
3. Open in a different window the file with the BIOS image
4. Copy the contents of the BIOS image file and paste, replacing, at the end of the card
5. Save the changes to the SD card

Linux

Follow these steps:

1. After finding which device belongs to the SD card (**sdb** for this example), execute **fdisk** to find out the size

```
fdisk -l /dev/sdb
Disk /dev/sdb: 59.49 GiB, 63864569856 bytes, 124735488 sectors
Disk model: SD/MMC
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
(...)
```

2. As the sector size is 512 bytes, subtract 16 (for this example, $124735488-16=124735472$) and use that info to inject the BIOS at the end:

```
sudo dd if=Next186_BIOS_zxdos_ddr3.COM of=/dev/sdb bs=512 seek=124735472
```

macOS

Steps to follow:

1. After finding out which disk is the card (**disk6** for this example), open **fdisk** to find out the size in sectors

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

Disk: /dev/rdisk6    geometry: 15566/255/63 [30535680 sectors]
(...)
Enter 'help' for information
fdisk: 1>q
```

2. As the sectors are 512 bytes in size, subtract 16 (for this example, $30535680-16=30535664$) and use

that number to inject the data at the end:

```
sudo diskutil unmountDisk /dev/disk6
sudo dd if=Next186_BIOS_zxdos_ddr3.COM of=/dev/rdisk6 bs=512 seek=30535664
```

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

Keyboard

Special keys and buttons

The following ZXUnGo+ keyboard combinations are obtained while in **PCXT** mode (**Caps Shift+Symbol Shift+F**, then **U** and then **0**). Please read [the section dedicated to ZXUnGo+ keyboard modes](#) for more information.

While the core is running:

- **Ctrl+Alt+Del** (**Caps Shift+Symbol Shift+F** and then **N** on ZXUnGo+): Soft Reset
- **Ctrl+Alt+Backspace** (**Caps Shift+Symbol Shift+F** and then **B** on ZXUnGo+): Hard reset. Backspace is the delete key, located in the top-right portion of the keyboard, above **Enter**

Basic Guide



1. Start the ZX-Uno
2. For a ZXUnGo+, change the keyboard mode to **PC XT** keyboard mode (**Caps Shift+Symbol Shift+U, F, and then 0**)
3. Reboot the ZX-Uno without losing the temporary keyboard mode (**Caps Shift+Symbol Shift+F** and then **B**)
4. Quickly, press **Caps Shift+2** or **Caps Shift+6** and select the PC XT core to boot
5. Ensure that the DOS keyboard configuration is in an spanish mode (**KEYB ES.**)



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

PC XT CGA

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

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

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

SD card format

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



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



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

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

Keyboard

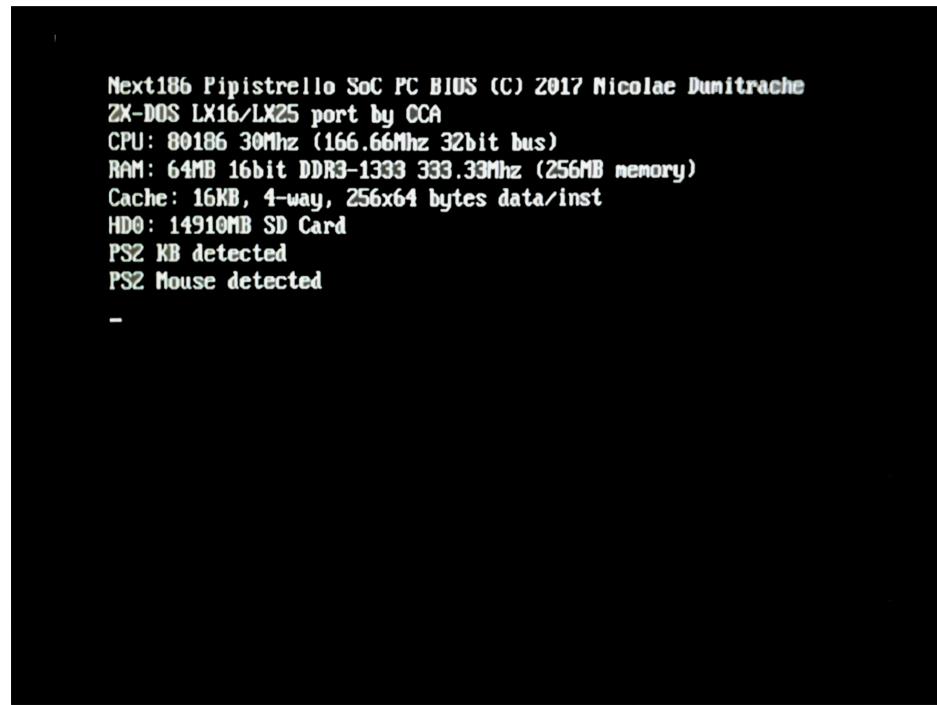
Special keys and buttons

The following ZXUnGo+ keyboard combinations are obtained while in **PCXT** mode (**Caps Shift+Symbol Shift+F**, then **U** and then **0**). Please read [the section dedicated to ZXUnGo+ keyboard modes](#) for more information.

While the core is running:

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

Basic Guide



1. Start the ZX-Uno
2. For a ZXUnGo+, change the keyboard mode to **PC XT** keyboard mode (**Caps Shift+Symbol Shift+U**, **F**, and then **0**)
3. Reboot the ZX-Uno without losing the temporary keyboard mode (**Caps Shift+Symbol Shift+F** and then **B**)
4. Quickly, press **Caps Shift+2** or **Caps Shift+6** and select the PC XT core to boot



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

Pong

Pong was [one of the earliest arcade video games](#) manufactured by Atari.

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

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

SD format

This core does not use the SD card.

Keyboard

Special keys and buttons

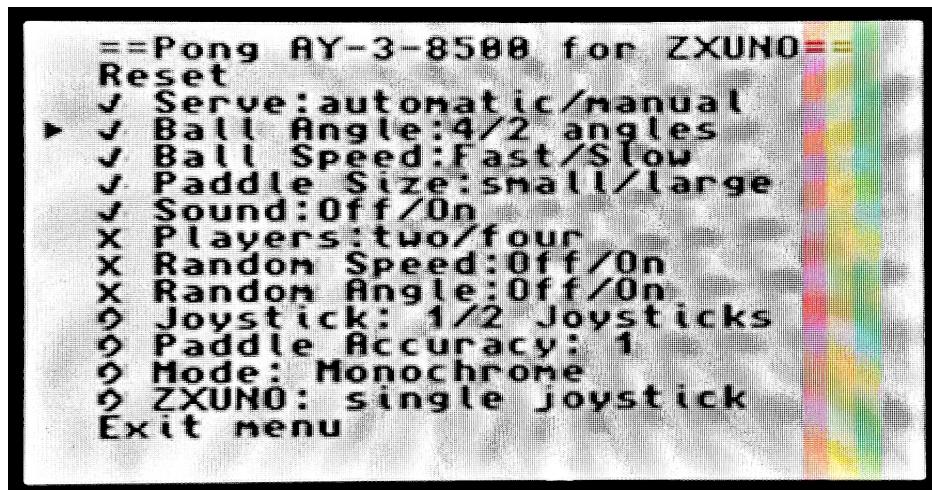
The following ZXUnGo+ keyboard combinations are obtained while in any keyboard mode except the normal [ZX](#) mode. Please read [the section dedicated to ZXUnGo+ keyboard modes](#) for more information.

While the core is running:

- **Esc** ([Caps Shift+Space](#) on ZXUnGo+) or joystick button 2: Show or hide configuration menu
- **Ctrl+Alt+Backspace** ([Caps Shift+Symbol Shift+F](#) and then **B** on ZXUnGo+) or **F11** ([Caps Shift+Symbol Shift+Q](#) on ZXUnGo+): Hard reset. Backspace is the delete key, located in the top-right portion of the keyboard, above **Enter**
- **F3** ([Caps Shift+Symbol Shift+3](#) on ZXUnGo+) or **F12** ([Caps Shift+Symbol Shift+W](#) on ZXUnGo+): Restart game
- Number between **1** and **7**: Change the game variant
- Joystick 2 (right): Control right pad (Player 1).
- Joystick 1 (left): Control left pad (Player 2)
- **Cursor up** and **Cursor down** ([Caps Shift+6](#) and [Caps Shift+7](#) on ZXUnGo+) or **0** and **K**: Control right pad (Player 1 in 2 player mode and player 3 in 4 player mode)
- **Q** and **A**: Control left pad (Player 2 in 2 player mode and player 4 in 4 player mode)
- **Z**, **M** or joystick button 1: Manual serve
- Cursor keys ([Caps Shift+6](#) and [Caps Shift+7](#) on ZXUnGo+) and **Enter** to use the menu

Basic Guide

Pressing **Esc** (**Caps Shift+Space** on ZXUnGo+) or joystick button 2 shows or hides the configuration menu. Cursor keys (**Caps Shift+6** and **Caps Shift+7** on ZXUnGo+) and **Enter** to select and choose menu options.



The following options are available:

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

SAM Coupé

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

The ZX-Uno cores has these features:

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

SD card format

This core does not use the SD card

Keyboard

Special keys and buttons

While the core is running:

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

Basic Guide



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



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

Sega Master System

The [Sega Master System](#) was a video game console manufactured by Sega.

The ZX-Uno core is based on [Ben's original work for Papilio Plus](#).

Some of the core features are:

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

SD card format

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

See the [corresponding section](#) for instructions of how to install the Sega Master System core in ZX-Uno.

Keyboard

Special keys and buttons

While the core is running:

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

Basic Guide

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



SmartROM

SmartROM is a kind of firmware that 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 [esxdos 0.8.8](#) to the card, and create a directory named **ZXUNO** with, at least, [the SMARTROM.ZX1 file](#) and a RomPack file named **ROMS.ZX1** inside.

Finally, if your PS/2 keyboard is not in Spanish, you can copy the corresponding [keyboard map file](#), renamed as **KEYMAP.ZX1**.

See the [corresponding section](#) for instructions of how to install the SmartROM core in ZX-Uno.

Vectrex

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

The ZX-Uno core is based on [DarFPGA`'s original work](#).

Some of its features are:

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

SD card format

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

At [archive.org](#) you can obtain software for Vectrex.

See the [corresponding section](#) for instructions of how to install the Vectrex core in ZX-Uno.

Keyboard

Special keys and buttons

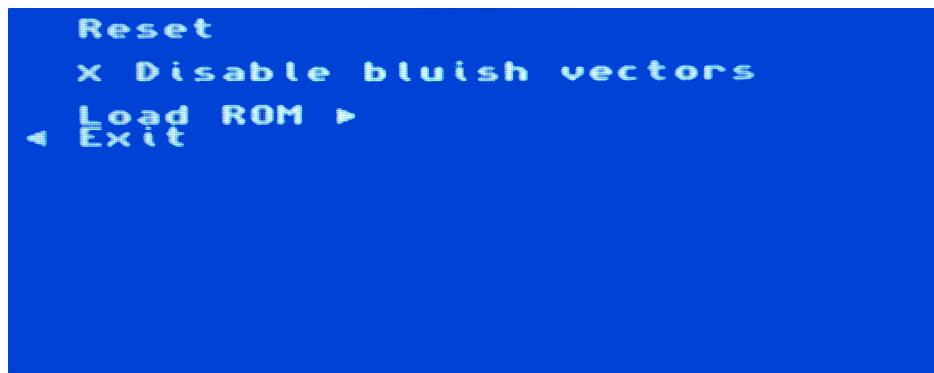
The following ZXUnGo+ keyboard combinations are obtained while in any keyboard mode except the normal **ZX** mode. Please read [the section dedicated to ZXUnGo+ keyboard modes](#) for more information.

During the core execution:

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

Basic guide

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



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

The following options are available:

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

Videopac

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

The ZX-Uno core is made by avlixa, and is based on ZX DOS core by yomboprime.

Some features of the core are:

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

SD card format

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

See the [corresponding section](#) for instructions of how to install the Videopac core in ZX-Uno.

Keyboard

Special keys and buttons

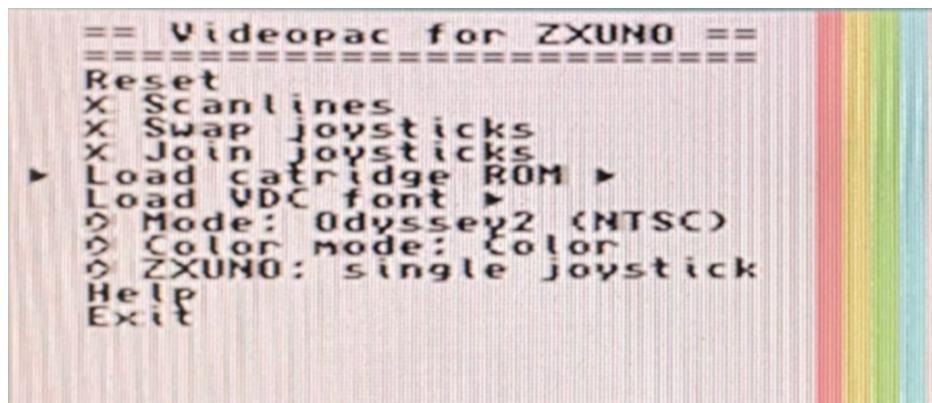
The following ZXUnGo+ keyboard combinations are obtained while in any keyboard mode except the normal ZX mode. Please read [the section dedicated to ZXUnGo+ keyboard modes](#) for more information.

During the core execution:

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

Basic Guide

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



The following options are available:

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



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



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



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

Change VDC ROM charset

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

Those files can be made following the instructions and using the editor available at the project repository, following [this link](#).

ZX81

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

The ZX-Uno version has been made by Jepalza based on a previous ZX97 implementation.

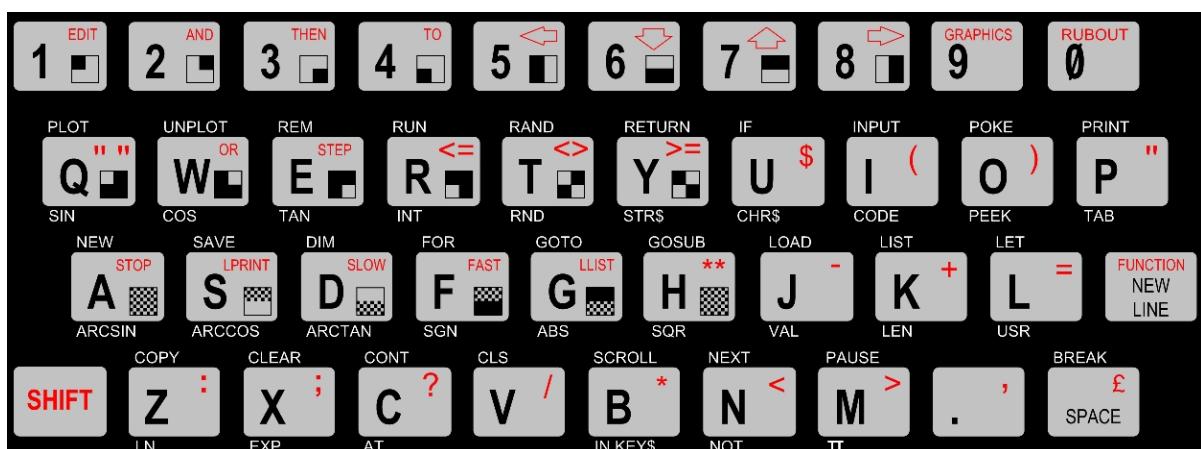
SD card format

This core does not use the SD card

Keyboard

The following ZXUnGo+ keyboard combinations are obtained while in ZX81 mode (**Caps Shift+Symbol Shift+F**, then **U** and then **F**). Please read [the section dedicated to ZXUnGo+ keyboard modes](#) for more information.

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



Basic Guide

You can load a external tape or from other external audio device with the command **LOAD""**. Take note, that, while loading, the video output is 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.

ZX81 and ZX80

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

Features:

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

SD card format

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

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

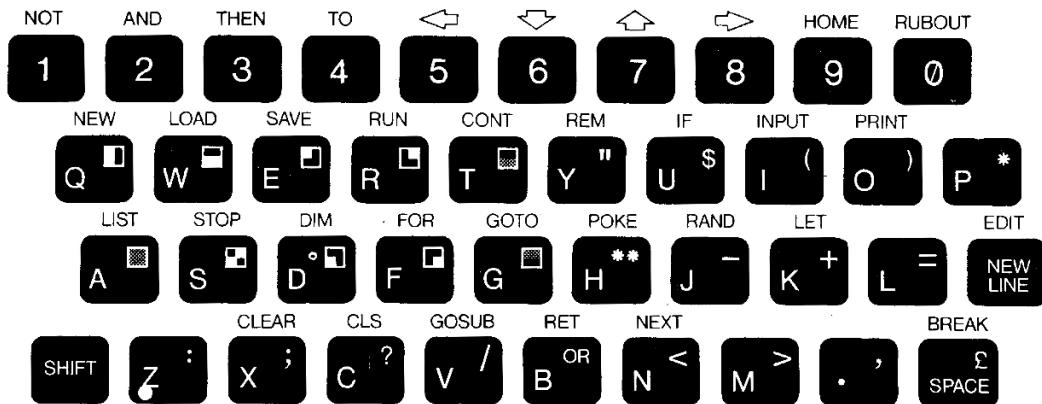
See the [corresponding section](#) for instructions of how to install the ZX81 and ZX80 core in ZX-Uno.

Keyboard

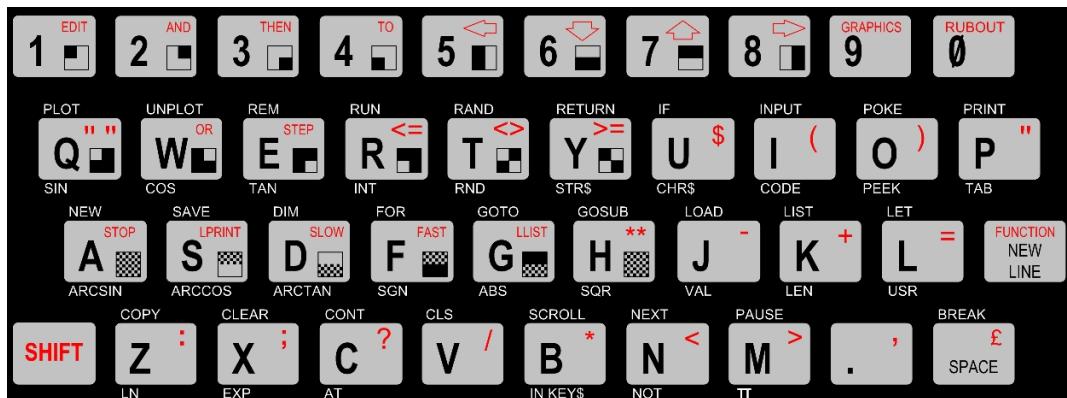
The following ZXUnGo+ keyboard combinations are obtained while in **ZX81** mode (**Caps Shift+Symbol Shift+F**, then **U** and then **F**). Please read [the section dedicated to ZXUnGo+ keyboard modes](#) for more information.

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

ZX80



ZX81



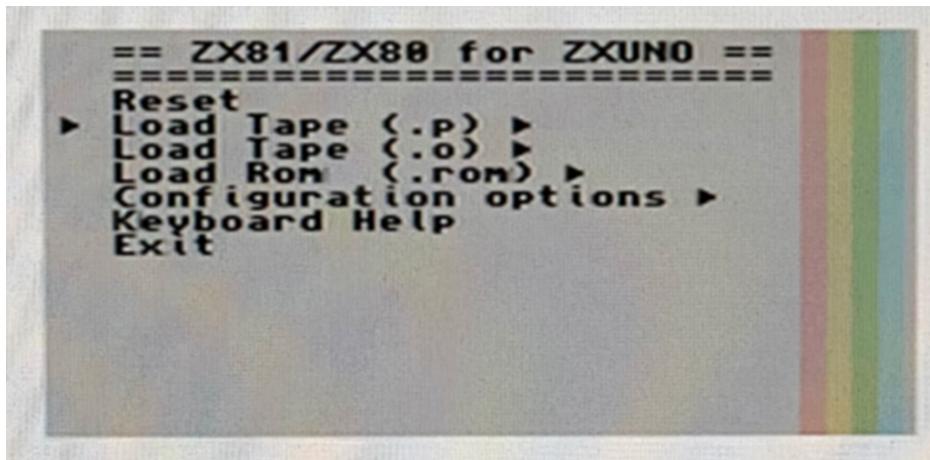
Special keys and buttons

During the core execution:

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

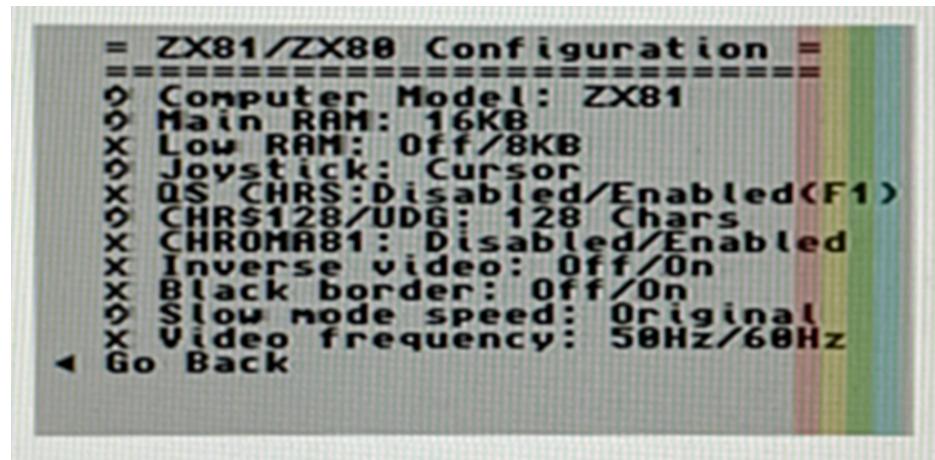
Basic Guide

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



The following options are available:

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



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

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



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

.p files with colorization and char are supported.

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



The recommended options for most games are:

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

Other Hardware

Joy2PS2

An interface to which one or two gamepads from various old platforms can be connected, and converts their events into keystrokes from a PS/2 port. This way, the gamepads can be used to any device that supports a keyboard.



The official website <http://spark2k06.strangled.net/joy2ps2/> and the firmware code repository is <https://github.com/spark2k06/zxuno/tree/master/joy2ps2>

On startup, the main gamepad is always mapped to cursor keys, the first button to **Enter** and the second one to **Escape**.

Pressing **Start+Button 1** or **Select+Start** or pressing and releasing the **Keymapper** key (for gamepads without **Start** button), enters into SHIFT mode, where you can enable several special actions:

	SHIFT MODE											JOY 2 PS2
ZX-UNO	x 1	x 2	x 3	x 4	x 5	x 6	x 7	x 8	x 9	x 10	x 11	
UP	KEY 'R'	KEY '0'	KEY '1'	KEY '2'	KEY '3'	KEY '4'	KEY '5'	KEY '6'	KEY '7'	KEY '8'	KEY '9'	
DOWN	RESET	MRESET	ROMS	CORES	BIOS							
LEFT	KEYB MODE	KEY 'ESC'	KEYMAP SELECT	VIDEO SELECT								
RIGHT	NMI	LOAD "" (128K)	LOAD "" (48K)									
B1+B2	DB15/DB9 DB9 x 2											
START+UP	KB SPEED UP	KB SPEED UP	KB SPEED UP									
START+DOWN	KB SPEED DOWN	KB SPEED DOWN	KB SPEED DOWN									
B1+LEFT	DISABLE HOST->KB											
B1+RIGHT	ENABLE HOST->KB											

For example, to reset a ZX-Uno, you have to press (while in SHIFT mode), 1 time down and then the button 1.

Firmware upgrade

To have the latest versions the flash memory of Joy2PS2 can be upgraded with the corresponding firmware version.

You need a computer (Windows, Mac, Linux), and AVR binary upgrade software like [XLoader for Windows](#) or [AVRDUDE](#).

You can download the latest firmware version from [the official repository](#).

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 [ZX-Uno audio input](#). It must have a 3.5 mm stereo jack on one side, and two mono outputs on the other side (one for each audio channel). The right audio mono is connected to the audio player.

Cassette Player

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

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

Computer

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

PlayTZX

This program for Windows, macOS or Linux, can play directly a [TZX](#) tape file through the audio output of the computer.

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

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

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

If everything works fine, you will see at the shell the name of the different tape data blocks, while the sound is played and the ZX-Uno 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 [playtx](#) will be created, which you can copy anywhere and then use. You can delete the compilation directory.

Mobile phone, tablet, MP3 player, etc.

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

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

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



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

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

Audio file conversion

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

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

tape2wav from [Fuse Utilities](#) is a command line utility that can export from **TZX** **PZX** and **TAP** to **WAV**.

pzx2wav in [PZX Tools](#) is another command line utility which exports to **WAV**.

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

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

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

[Lynx2Wav](#) is a program that can convert Computers Lynx **TAP** files to **WAV** audio.

[2lynx2wav](#) also can convert Computers Lynx **TAP** files to audio files.

[cgc2wav](#) can convert EACA Colour Genie **CAS** files to **WAV** audio.

[castool](#), available with [MAME](#) can convert many kinds of tape file data to **WAV** audio.

[mzf2wav](#) converts **MZF** tape files to **WAV**.

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 extracted file from the SD, you can "clean" it leaving only the Spectrum core and the first Spectrum ROM with a command like this:

```
... zx123_tool.py -i FLASH.ZX1 -w -o FLASHempty.ZX1
```

Where **FLASH.ZX1** is the path to the original file and **FLASHempty.ZX1** is the path to the new "clean" file.

List the contents of an image

To see the contents of an image file named **FLASH.ZX1** (installed cores and some configuration info), you can use the command

```
... zx123_tool.py -i FLASH.ZX1 -l
```

To show the contents of the same file, including ZX Spectrum ROMs info:

```
... zx123_tool.py -i FLASH.ZX1 -l -r
```

Change the BIOS of an image

To change the BIOS inside a file named **FLASH.ZX1**, using the BIOS file named **FIRMWARE.ZX1**

```
...zx123_tool.py -i FLASH.ZX1 -a BIOS,FIRMWARE.ZX1
```

You can, at the same time, modify some of the default parameters. For example, with the options; **-m** for video mode: 0 (PAL), 1 (NTSC) or 2 (VGA), **-k** for the keyboard layout: 0 (Auto), 1 (ES), 2 (EN) or 3 (Spectrum).

This way to change the BIOS of a file named **FLASH.ZX1**, using the BIOS file **FIRMWARE.ZX1**, and also set the video mode to VGA:

```
...zx123_tool.py -i FLASH.ZX1 -a BIOS,FIRMWARE.ZX1 -m 2
```

There are also options to set the BIOS boot delay time, the default core or the default Spectrum ROM. See the [documentation](#) for more info.

Add a Spectrum ROM to an image

To add a Spectrum ROM file named `48.rom`, with the name `Spec48` and using the slot number 5, you can use a command like:

```
...zx123_tool.py -i FLASH.ZX1 -a ROM,5,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 using slot number 5, install a core at space 3, from a file named **TBBLUE.ZX1**, with the name **TBBlue**, as default core:

```
... zx123_tool.py -i FLASH.ZX1 -w -o FLASHnew.ZX1 -a BIOS,FIRMWARE.ZX1 -m 2 -k 3 -a ROM,5,xdnlh17,Spec48,48.rom -a CORE,3,TBBlue,TBBLUE.ZX1 -c 3
```

Firmware recovery

Sometimes (e.g. when installing an experimental core or when upgrading the ZX Spectrum Core or the BIOS) it may happen that the ZX-Uno 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.

Recovery using audio

If a test pattern is shown when booting a ZX-Uno with a joystick connected, and having both the fire and up buttons pressed, then the ZX-Uno is waiting to load a BIOS using audio input.

Hardware required:

- ZX-Uno with joystick connector
- A compatible joystick or gamepad
- Audio cable and some way to [load from tape](#) that can read **.TAP** files
- SD Card

Software required:

- A **firmware.tap** file, (like the one available [here](#))
- A **FIRMWARE.ZX1** file (remember that the latest version of the firmware files can be downloaded from [the official repository](#)), that you have to copy to the root of the SD Card

The **firmware.tap** file can be made using this command



GenRom sm12 BIOS FIRMWARE.ZX1 firmware.tap

The GenRom utility can be obtained from [the official repository](#) (Windows version, [available here](#)).

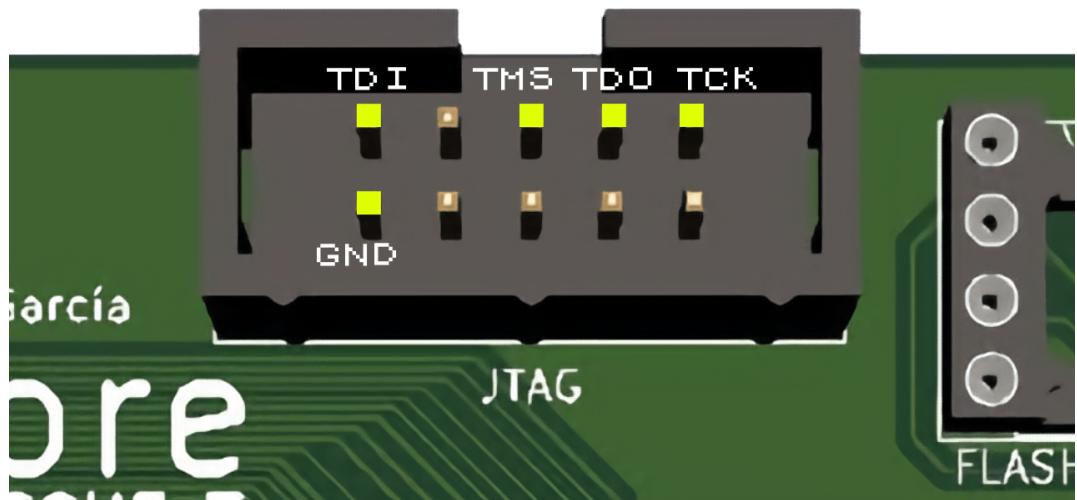
Instruction Steps:

1. Insert in the ZX-Uno the SD card and boot the ZX-Uno Arrancar el ZX-Uno while having both the fire and up buttons pressed
2. Play the **firmware.tap** file
3. Be very attentive, and quickly press **F2** (**Caps Shift+1** on ZXUnGo+) when the load finishes to access the BIOS (if this fails, the whole process has to be started again from the beginning).
4. Once in the BIOS, select **Upgrade**, choose "*Upgrade BIOS for ZX*", and then "*SDfile*". The system will read the file **FIRMWARE…** and notify when finished.

If the process finishes successfully, reboot the ZX-Uno and then it will be possible to use any of the normal methods to [update ROMs, Cores, flash memory, etc..](#)

JTAG cable connections

Later, in some of the recovery steps, when talking about jump wires or USB-Blaster connections to the ZXUnCore board, you can use these images as reference.



DO NOT connect the 3V line



ZXUnCore has the right pinout to use directly a USB-Blaster 2x5 connector.

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
- SD for ZX-Uno with the first partition formatted as FAT16 or FAT32
- Keyboard and display for ZX-Uno

Software required:

- Flash image and recovery file for ZX-Uno from [Github repository](#) and [ZX-Uno forum](#)

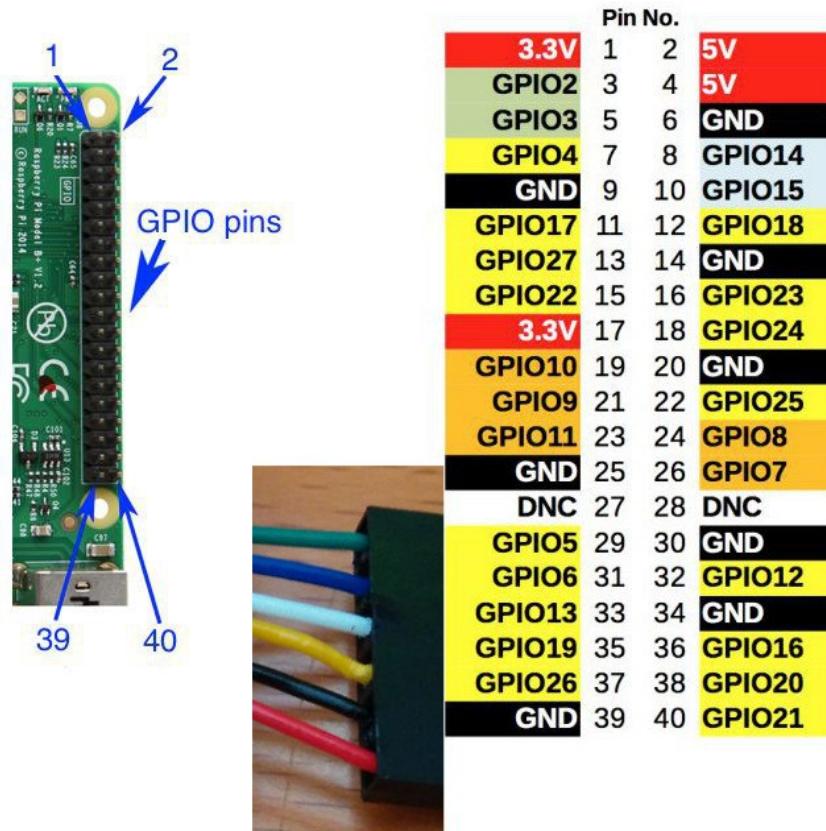
Instruction Steps:

1. Install Raspberry Pi OS (formerly known as Raspbian) to the Raspberry Pi SD card (using [the official download](#), [NOOBS](#), [PINN](#), etc.)
2. Install Open OCD:

```
sudo apt-get update
sudo apt-get install git autoconf libtool make pkg-config
sudo apt-get install libusb-1.0-0 libusb-1.0-0-dev telnet
sudo apt-get install libusb-dev libftdi-dev
git clone git://git.code.sf.net/p/openocd/code openocd-code
cd openocd-code/
./bootstrap
./configure --enable-usb_blaster --enable-sysfsgpio --enable-bcm2835gpio
make
sudo make install
cd ..
rm -rf ./openocd-code
```

3. Connect USB-Blaster or jump wires if using GPIO. In this case, open the ZX-Uno 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:

ZX-Uno 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/zxuncore/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/zxuncore/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 ZX-Uno.

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:

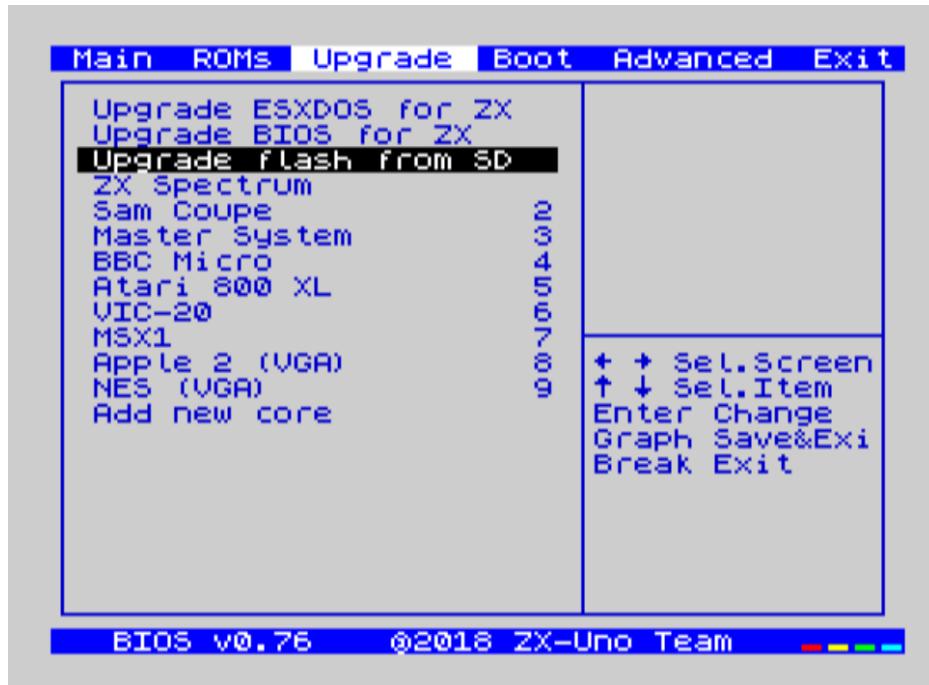
```
cd /home/pi/zxuncore/unbrick  
sudo openocd -f /home/pi/zxuncore/unbrick/raspberrypi2-native.cfg -f  
/usr/local/share/openocd/scripts/cpld/xilinx-xc6s.cfg -c "init; xc6s_program xc6s.tap;  
pld load 0 recovery.bit ; exit"
```

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 ZX-Uno the SD card formatted as FAT16 o FAT32, and with the [FLASH.ZX1](#) file downloaded previously.
13. If using a USB-Blaster connection, unplug the connector.

14. Select the option **Upgrade Flash from SD**. Press Enter, choose **Yes**, and press Enter again to start the Flash writing process.



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



Usually, the recovery image is set to use a PS/2 keyboard so, for a ZXUnGo+, some key combinations, like **Caps Shift + 5**, etc may not work. In this case, you have to change the keyboard to any mode except the normal **ZX** (please read [the section dedicated to ZXUnGo+ keyboard modes](#) for more information) to make them work temporarily.

15. After some minutes, the process will end, and, after turning the ZX-Uno off and on, it should start fine.



If no image is shown, press again **Scroll Lock** (**Caps Shift+Symbol Shift+Y** on ZXUnGO+) to switch between RGB and VGA modes. In this case, you should have to enter the BIOS and change [the right advanced setting](#) that matches your display.

For a ZXUnGo+, since the recovery image uses a PS/2 configuration as default, follow this steps to set up the BIOS correctly:



1. If you see no image, switch between RGB and VGA mode (**Caps Shift+Symbol Shift+Y**)
2. Change to any keyboard mode except the normal **ZX**. Please read [the section dedicated to ZXUnGo+ keyboard modes](#) for more information
3. Reboot ZXUnGo+ without losing the temporary keyboard mode (**Caps Shift+Symbol Shift+F** and then **B**)

4. Quickly, press **Caps Shift + 1**
5. Again, if there's no image, switch between RGB and VGA mode (**Caps Shift+Symbol Shift+Y**)
6. Navigate through BIOS and turn on these options:
 - **Advanced** → **Keyboard Layout: Spectrum**
 - **Advanced** → **Video: VGA** (only if there was no image)
7. Save changes:
 - **Exit** → **Save changes and exit**
8. Completely turn off the ZX-Uno and turn it on again



If no image is shown, press again **Scroll Lock** (**Caps Shift+Symbol Shift+Y** on ZXUnGo+) to switch between RGB and VGA modes. In this case, you should have to enter the BIOS and change **the right advanced setting** that matches your display.

Recovery using macOS and USB-Blaster cable

Hardware required:

- USB-Blaster cable. Using the right pins for ZXUnCore, as explained earlier
- Flash image and recovery file for ZX-Uno from [Github repository](#) and [ZX-Uno forum](#)

Software required:

- Mac OS
- Extra data folder for UrJTAG, obtained from [here](#)
- [Homebrew for Mac OS](#)
- UrJTAG: following instructions from [here](#):

Instruction Steps:

1. Prepare UrJTAG instal:

```
brew install libftdi libusb pkg-config  
git clone https://github.com/C-Elegans/urjtag.git  
cd urjtag
```

2. Copy the extra data folder for UrJTAG, into urjtag **data** folder.
3. Start compiling:

```
./configure --with-libftdi --with-libusb --with-ftd2xx --with-inpout32 --enable-  
python=no  
make -j4  
sudo make install
```

4. Copy **FLASH.ZX1** file to the root of the ZX-Uno SD card.
5. Connect USB-Blaster cable to ZX-Uno and the Mac
6. Turn on the ZX-Uno.
7. Make sure that, we are in the directory where **recovery.bit** is, and execute **jtag** command.

8. A new shell appears. Now type the commands:

```
cable usbblaster  
detect  
pld load recovery.bit
```

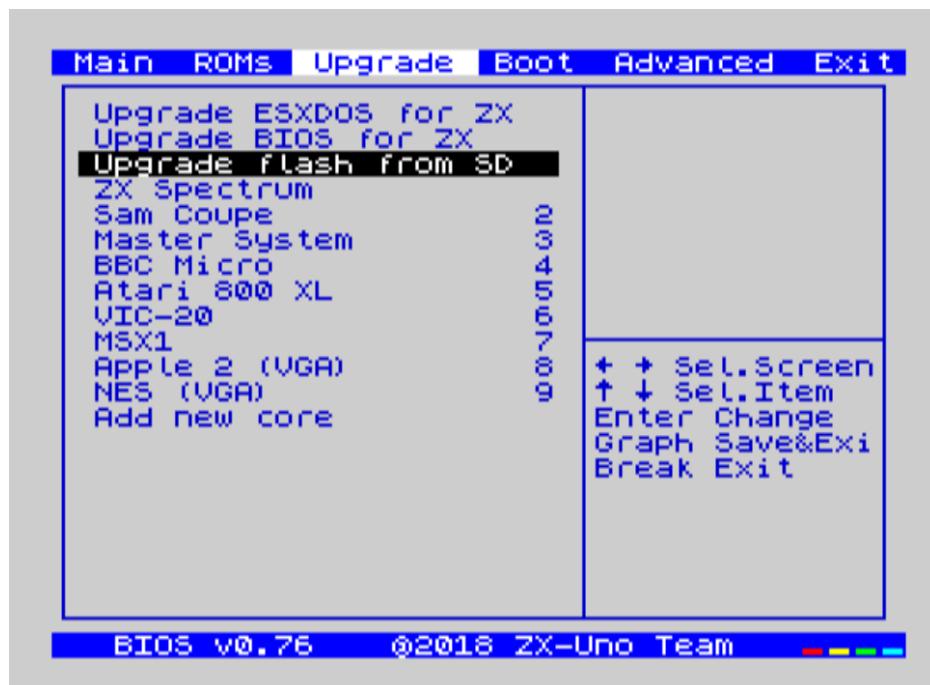


Make sure, when using `detect`, that the device is shown as detected. You may need to execute `detect` several times until it appears.

```
UrJTAG 2018.06 #  
Copyright (C) 2002, 2003 ETC s.r.o.  
Copyright (C) 2007, 2008, 2009 Kolja Waschk and the respective authors  
  
UrJTAG is free software, covered by the GNU General Public License, and you are  
welcome to change it and/or distribute copies of it under certain conditions.  
There is absolutely no warranty for UrJTAG.  
  
warning: UrJTAG may damage your hardware!  
Type "quit" to exit, "help" for help.  
  
jtag> cable usbblaster  
Connected to libftdi driver.  
jtag> detect  
IR length: 6  
Chain length: 1  
Device Id: 00100100000000000100000010010011 (0x24004093)  
    Manufacturer: Xilinx (0x093)  
    Part(0):      xc6slx25 (0x4004)  
    Stepping:     2  
    Filename:     /usr/local/share/urjtag/xilinx/xc6slx25/xc6slx25  
jtag> pld load recovery.zxd.bit  
Bitstream information:  
    Design: tld_zxuno_lx16.ncd;UserID=0xFFFFFFFF  
    Part name: 6slx25ftg256  
    Date: 2020/05/29  
    Time: 01:32:07  
    Bitstream length: 801462  
jtag> █
```

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

If there is no image on the display, press **Scroll Lock** (**Caps Shift+Symbol Shift+Y** on ZXUnGo+): to switch between RGB and VGA modes, just in case the recovery BIOS did start in the wrong mode for our setup.



10. Insert in the ZX-Uno the SD card formatted as FAT16 o FAT32, and with the [FLASH.ZX1](#) file downloaded previously.
11. Unplug the USB-Blaster.

12. Select the option **Upgrade Flash from SD**. Press Enter, choose **Yes**, and press Enter again to start the Flash writing process.



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



Usually, the recovery image is set to use a PS/2 keyboard so, for a ZXUnGo+, some key combinations, like **Caps Shift + 5**, etc may not work. In this case, you have to change the keyboard to any mode except the normal **ZX** (please read [the section dedicated to ZXUnGo+ keyboard modes](#) for more information), to make them work temporarily.

13. After some minutes, the process will end, and, after turning the ZX-Uno off and on, it should start fine.



If no image is shown, press again **Scroll Lock** (**Caps Shift+Symbol Shift+Y** on ZXUnGo+): to switch between RGB and VGA modes. In this case, you should have to enter the BIOS and change [the right advanced setting](#) that matches your display.

For a ZXUnGo+, since the recovery image uses a PS/2 configuration as default, follow this steps to set up the BIOS correctly:



1. If you see no image, switch between RGB and VGA mode (**Caps Shift+Symbol Shift+Y**)
2. Change to any mode except the normal **ZX**. Please read [the section dedicated to ZXUnGo+ keyboard modes](#) for more information.
3. Reboot the ZXUnGo+ without losing the temporary keyboard mode (**Caps Shift+Symbol Shift+F** and then **B**)

4. Quickly, press **Caps Shift + 1**
5. Again, if there's no image, switch between RGB and VGA mode (**Caps Shift+Symbol Shift+Y**)
6. Navigate through BIOS and turn on these options:
 - **Advanced** → **Keyboard Layout: Spectrum**
 - **Advanced** → **Video: VGA** (only if there was no image)
7. Save changes:
 - **Exit** → **Save changes and exit**
8. Completely turn off the ZX-Uno and turn it on again



When using Linux with **urjtag**, the process should be quite similar, although the dependencies (`libftdi` `libusb` `pkg-config`) would have to be installed with the appropriate package manager (apt, yum, pacman, etc.)

Recovery using Windows and USB-Blaster cable

Hardware required:

- USB-Blaster cable. Using the right pins for ZXUnCore, [as explained earlier](#)
- Flash Image file and recovery file for ZX-Uno. The same ones as for Raspberry Pi, from the [GitHub repository](#) and [ZX-Uno forum](#)

Software required:

- Windows
- USB-Blaster drivers for Windows, available at [ZX-Uno forums](#)
- UrJTAG, for Windows, available at [the official repository](#)

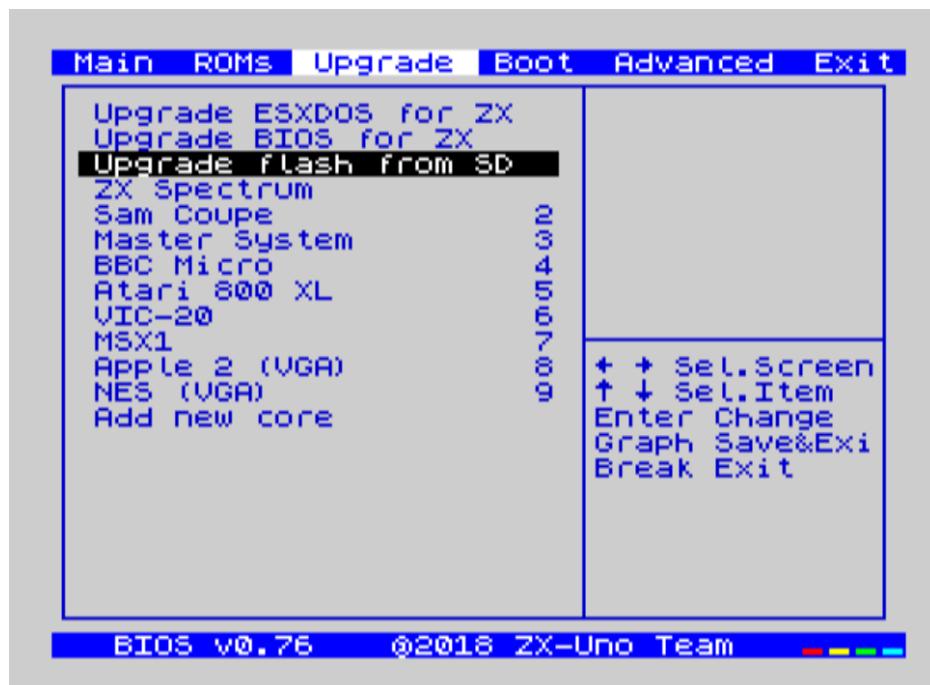
Instruction Steps:

1. Extract the ZIP file with drivers (obtained from the [ZX-Uno forums](#))
2. Plug USB-Blaster to the Windows PC and install the driver choosing to select manually installation files and using the folder [drivers](#) obtained after extracting the ZIP
3. Extract UrJTAG Windows software, obtained at the [official repository](#)
4. Copy [FLASH.ZX1](#) file to the root of the ZX-Uno SD card.
5. Connect USB-Blaster cable to ZXUnCore and the PC
6. Make sure that you are in the directory where [recovery.bit](#) is, and execute [jtag.exe](#) command.
7. Turn on the ZX-Uno.
8. A new shell appears. Now type the commands:

```
cable usbblaster  
detect  
pld load recovery.bit
```

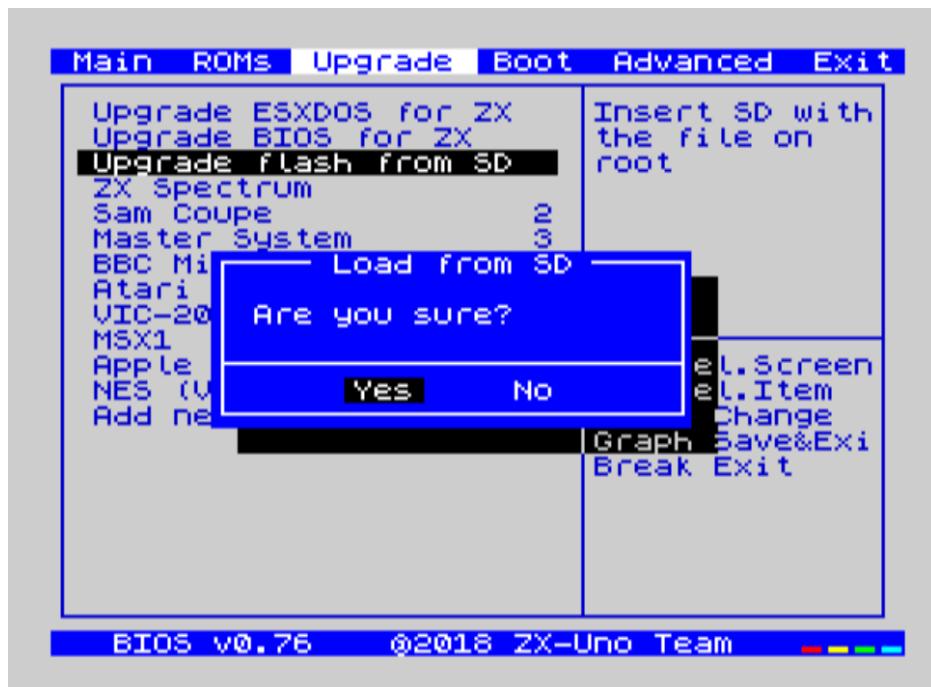
9. If all goes well, we will see that the FPGA LED change their state and the BIOS is shown on the display.

If there is no image on the display, press **Scroll Lock** (**Caps Shift+Symbol Shift+Y** on ZXUnGo+): to switch between RGB and VGA modes, just in case the recovery BIOS did start in the wrong mode for our setup.



10. Insert in the ZX-Uno the SD card formatted as FAT16 or FAT32, and with the [FLASH.ZX1](#) file downloaded previously.
11. Unplug the USB-Blaster.

12. Select the option **Upgrade Flash from SD**. Press Enter, choose **Yes**, and press Enter again to start the Flash writing process.



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



Usually, the recovery image is set to use a PS/2 keyboard so, for a ZXUnGo+, some key combinations, like **Caps Shift + 5**, etc may not work. In this case, you have to change to keyboard any mode except the normal **ZX**. Please read [the section dedicated to ZXUnGo+ keyboard modes](#) for more information.

13. After some minutes, the process will end, and, after turning the ZX-Uno off and on, it should start fine.



If no image is shown, press again **Scroll Lock** (**Caps Shift+Symbol Shift+Y** on ZXUnGo+): to switch between RGB and VGA modes. In this case, you should have to enter the BIOS and change [the right advanced setting](#) that matches your display.

For a ZXUnGo+, since the recovery image uses a PS/2 configuration as default, follow this steps to set up the BIOS correctly:



1. If you see no image, switch between RGB and VGA mode (**Caps Shift+Symbol Shift+Y**)
2. Change to any keyboard mode except the normal **ZX**. Please read [the section dedicated to ZXUnGo+ keyboard modes](#) for more information.
3. Reboot the ZXUnGo+ without losing the temporary keyboard mode (**Caps Shift+Symbol Shift+F** and then **B**)
4. Quickly, press **Caps Shift + 1**

5. Again, if there's no image, switch between RGB and VGA mode (**Caps Shift+Symbol Shift+Y**)
6. Navigate through BIOS and turn on these options:
 - **Advanced → Keyboard Layout: Spectrum**
 - **Advanced → Video: VGA** (only if there was no image)
7. Save changes:
 - **Exit → Save changes and exit**
8. Completely turn off the Zx-Uno and turn it on again

References

Scan Codes

101-, 102-, and 104-key Scan Codes

KEY	MAKE	BREAK
A	1C	F0,1C
B	32	F0,32
C	21	F0,21
D	23	F0,23
E	24	F0,24
F	2B	F0,2B
G	34	F0,34
H	33	F0,33
I	43	F0,43
J	3B	F0,3B
K	42	F0,42
L	4B	F0,4B
M	3A	F0,3A
N	31	F0,31
O	44	F0,44
P	4D	F0,4D
Q	15	F0,15
R	2D	F0,2D
S	1B	F0,1B
T	2C	F0,2C
U	3C	F0,3C
V	2A	F0,2A
W	1D	F0,1D
X	22	F0,22
Y	35	F0,35
Z	1A	F0,1A
0	45	F0,45
1	16	F0,16
2	1E	F0,1E
3	26	F0,26
4	25	F0,25
5	2E	F0,2E
6	36	F0,36
7	3D	F0,3D
8	3E	F0,3E
9	46	F0,46
'	0E	F0,0E
-	4E	F0,4E
=	55	F0,55
\	5D	F0,5D
BKSP	66	F0,66
SPACE	29	F0,29
TAB	0D	F0,0D
CAPS	58	F0,58
L SHFT	12	F0,12
L CTRL	14	F0,14
L GUI	E0,1F	E0,F0,1F
L ALT	11	F0,11
R SHFT	59	F0,59
R CTRL	E0,14	E0,F0,14
R GUI	E0,27	E0,F0,27
R ALT	E0,11	E0,F0,11
APPS	E0,2F	E0,F0,2F
ENTER	5A	F0,5A
ESC	76	F0,76
F1	5	F0,05
F2	6	F0,06
F3	4	F0,04
F4	0C	F0,0C
F5	3	F0,03
F6	0B	F0,0B
F7	83	F0,83
F8	0A	F0,0A
F9	1	F0,01
F10	9	F0,09
F11	78	F0,78
F12	7	F0,07
PRNT SCRN	E0,12, E0,7C	E0,F0, 7C,E0, F0,12
SCROLL	7E	F0,7E
PAUSE	E1,14,77, E1,F0,14,F0,77	-NONE-

101-, 102-, and 104-key Scan Codes

KEY	MAKE	BREAK
[54	F0,54
INSERT	E0,70	E0,F0,70
HOME	E0,6C	E0,F0,6C
PG UP	E0,7D	E0,F0,7D
DELETE	E0,71	E0,F0,71
END	E0,69	E0,F0,69
PG DN	E0,7A	E0,F0,7A
U ARROW	E0,75	E0,F0,75
L ARROW	E0,6B	E0,F0,6B
D ARROW	E0,72	E0,F0,72
R ARROW	E0,74	E0,F0,74
NUM	77	F0,77
KP /	E0,4A	E0,F0,4A
KP *	7C	F0,7C
KP -	7B	F0,7B
KP +	79	F0,79
KP EN	E0,5A	E0,F0,5A
KP .	71	F0,71
KP 0	70	F0,70
KP 1	69	F0,69
KP 2	72	F0,72
KP 3	7A	F0,7A
KP 4	6B	F0,6B
KP 5	73	F0,73
KP 6	74	F0,74
KP 7	6C	F0,6C
KP 8	75	F0,75
KP 9	7D	F0,7D
]	5B	F0,5B
:	4C	F0,4C
,	52	F0,52
.	41	F0,41
/	49	F0,49
	4A	F0,4A

Windows Multimedia Scan Codes

Key	Make Code	Break Code
Next Track	E0, 4D	E0, F0, 4D
Previous Track	E0, 15	E0, F0, 15
Stop	E0, 3B	E0, F0, 3B
Play/Pause	E0, 34	E0, F0, 34
Mute	E0, 23	E0, F0, 23
Volume Up	E0, 32	E0, F0, 32
Volume Down	E0, 21	E0, F0, 21
Media Select	E0, 50	E0, F0, 50
E-Mail	E0, 48	E0, F0, 48
Calculator	E0, 2B	E0, F0, 2B
My Computer	E0, 40	E0, F0, 40
WWW Search	E0, 10	E0, F0, 10
WWW Home	E0, 3A	E0, F0, 3A
WWW Back	E0, 38	E0, F0, 38
WWW Forward	E0, 30	E0, F0, 30
WWW Stop	E0, 28	E0, F0, 28
WWW Refresh	E0, 20	E0, F0, 20
WWW Favorites	E0, 18	E0, F0, 18

ACPI Scan Codes

Key	Make Code	Break Code
Power	E0, 37	E0, F0, 37
Sleep	E0, 3F	E0, F0, 3F
Wake	E0, 5E	E0, F0, 5E

References

Spectrum

Scan Codes

101-, 102-, and 104-key Scan Codes

KEY	MAKE	BREAK
A	1C	F0,1C
B	32	F0,32
C	21	F0,21
D	23	F0,23
E	24	F0,24
F	2B	F0,2B
G	34	F0,34
H	33	F0,33
I	43	F0,43
J	3B	F0,3B
K	42	F0,42
L	4B	F0,4B
M	3A	F0,3A
N	31	F0,31
O	44	F0,44
P	4D	F0,4D
Q	15	F0,15
R	2D	F0,2D
S	1B	F0,1B
T	2C	F0,2C
U	3C	F0,3C
V	2A	F0,2A
W	1D	F0,1D
X	22	F0,22
Y	35	F0,35
Z	1A	F0,1A
0	45	F0,45
1	16	F0,16
2	1E	F0,1E
3	26	F0,26
4	25	F0,25
5	2E	F0,2E
6	36	F0,36
7	3D	F0,3D
8	3E	F0,3E
9	46	F0,46
.	0E	F0,0E
-	4E	F0,4E
=	55	F0,55
\	5D	F0,5D
BKSP	66	F0,66
SPACE	29	F0,29
TAB	0D	F0,0D
CAPS	58	F0,58
L SHFT	12	F0,12
L CTRL	14	F0,14
L GUI	E0,1F	E0,F0,1F
L ALT	11	F0,11
R SHFT	59	F0,59
R CTRL	E0,14	E0,F0,14
R GUI	E0,27	E0,F0,27
R ALT	E0,11	E0,F0,11
APPS	E0,2F	E0,F0,2F
ENTER	5A	F0,5A
ESC	76	F0,76
F1	5	F0,05
F2	6	F0,06
F3	4	F0,04
F4	0C	F0,0C
F5	3	F0,03
F6	0B	F0,0B
F7	83	F0,83
F8	0A	F0,0A
F9	1	F0,01
F10	9	F0,09
F11	78	F0,78
F12	7	F0,07
PRNT SCRN	E0,12, E0,7C	E0,F0, 7C,E0, F0,12
SCROLL	7E	F0,7E
PAUSE	E1,14,77, E1,F0,14,F0,77	-NONE-

101-, 102-, and 104-key Scan Codes

KEY	MAKE	BREAK
[54	F0,54
INSERT	E0,70	E0,F0,70
HOME	E0,6C	E0,F0,6C
PG UP	E0,7D	E0,F0,7D
DELETE	E0,71	E0,F0,71
END	E0,69	E0,F0,69
PG DN	E0,7A	E0,F0,7A
U ARROW	E0,75	E0,F0,75
L ARROW	E0,6B	E0,F0,6B
D ARROW	E0,72	E0,F0,72
R ARROW	E0,74	E0,F0,74
NUM	77	F0,77
KP /	E0,4A	E0,F0,4A
KP *	7C	F0,7C
KP -	7B	F0,7B
KP +	79	F0,79
KP EN	E0,5A	E0,F0,5A
KP .	71	F0,71
KP 0	70	F0,70
KP 1	69	F0,69
KP 2	72	F0,72
KP 3	7A	F0,7A
KP 4	6B	F0,6B
KP 5	73	F0,73
KP 6	74	F0,74
KP 7	6C	F0,6C
KP 8	75	F0,75
KP 9	7D	F0,7D
]	5B	F0,5B
;	4C	F0,4C
,	52	F0,52
.	41	F0,41
.	49	F0,49
/	4A	F0,4A

Windows Multimedia Scan Codes

Key	Make Code	Break Code
Next Track	E0, 4D	E0, F0, 4D
Previous Track	E0, 15	E0, F0, 15
Stop	E0, 3B	E0, F0, 3B
Play/Pause	E0, 34	E0, F0, 34
Mute	E0, 23	E0, F0, 23
Volume Up	E0, 32	E0, F0, 32
Volume Down	E0, 21	E0, F0, 21
Media Select	E0, 50	E0, F0, 50
E-Mail	E0, 48	E0, F0, 48
Calculator	E0, 2B	E0, F0, 2B
My Computer	E0, 40	E0, F0, 40
WWW Search	E0, 10	E0, F0, 10
WWW Home	E0, 3A	E0, F0, 3A
WWW Back	E0, 38	E0, F0, 38
WWW Forward	E0, 30	E0, F0, 30
WWW Stop	E0, 28	E0, F0, 28
WWW Refresh	E0, 20	E0, F0, 20
WWW Favorites	E0, 18	E0, F0, 18

ACPI Scan Codes

Key	Make Code	Break Code
Power	E0, 37	E0, F0, 37
Sleep	E0, 3F	E0, F0, 3F
Wake	E0, 5E	E0, F0, 5E

ZX-Uno control I/O registers

In the Spectrum core the **\$FC3B** and **\$FD3B** ports are reserved and assigned by the [ZXI committee](#). A total of 256 different I/O registers unique to the ZX-Uno family are accessed through these ports.

Port **\$FC3B (64571)** stores the address (**\$00 - \$FF**) of the I/O register to be accessed. It can be read to find out which register address was last allocated.

Port **\$FD3B (64827)** is the access port to the register selected with the previous port. Its meaning (read/write) depends on the implementation of each register.

For example, to allocate bank 16 of the SRAM to the address space **\$C000 - \$FFFFFF** during boot mode, using the **MASTERMAPPER** register, would be as follows:

```
ld bc,$fc3b      ;Port to set register number to use
ld a,1           ;Register $01 (MASTERMAPPER)
out (c),a        ;Selected. From now on, any access to $FD3B is using MASTERMAPPER
inc b            ;Register access port ($FD3B, just increment B)
ld a,16          ;SRAM Bank 16
out (c),a        ;Write to the MASTERMAPPER register
```

The registers implemented in the ZX-Uno family are described below.

\$00 MASTERCONF

Direction	Value after user reset	Value after master reset	Value after poweron
Read/Write	Does not change	00000001	00000001

Binary format (fields in **bold** can only be altered when LOCK=0):

LOCK	MODE1	DISCONT	MODE0	I2KB	DISNMI	DIVEN	BOOTM
------	-------	---------	-------	------	--------	-------	-------

- **BOOTM**: 1 indicates that the ZX-Uno is in boot mode (configuration mode). Boot mode only makes sense while the boot firmware is running, where some aspects of the ZX-Uno are allowed to be configured before going into run mode. MASTERCONF can always be read, both in boot mode and in run mode. It is set to 0 manually per program, at which point ZX-Uno enters run mode.
- **DIVEN**: set to 0 indicates that DIVMMC is not enabled in the system, although the SPI interface access ports of the SD/MMC slot are still available. The memory used by DIVMMC remains available for other uses. 1 indicates that DIVMMC is enabled. If enabled, an ESXDOS image must be loaded into the corresponding RAM bank before entering run mode. The default value of this bit is 0.
- **DISNMI**: set to 1 indicates that the DIVMMC NMI function will not be available. NMI will work, but will not cause ESXDOS to automate, thus leaving NMI control to the main system ROM. Bit added to improve DIVMMC compatibility with SE Basic IV. Defaults to 0 (ESXDOS handles NMI events).
- **I2KB**: a1 sets the ULA to return a value consistent with a Spectrum issue 2 when reading the keyboard port (\$FE). When 0, the returned value is compatible with issue 3 and later. It defaults to 0.
- **MODE1,MODE0**: specifies the ULA timing mode to accommodate different Spectrum models. 00 = ULA ZX Spectrum 48K PAL, 01 = ZX Spectrum 128K/+2 grey, 10 = Pentagon 128, 11 = 48K NTSC (262 scans).
- **DISCONT**: indicates whether memory contention should occur in the video memory. 0 to enable contention (48K and 128K compatibility). 1 to disable contention (Pentagon 128 compatibility).
- **LOCK**: When set to 1, it prevents further changes to certain bits in the MASTERCONF register, and also prevents access to the SPI Flash. This bit is reset to 0 only by a master reset (Ctrl-Alt-BkSpace) or by powering the clone off and on.

\$01 MASTERMAPPER

Direction	Value after user reset	Value after master reset	Value after poweron
Read/Write	Does not change	00000000	00000000

Only the lower 5 bits (values \$00 to \$1F) of this register are used. The value stored is the number of a 16KB bank of SRAM that will be paged at addresses \$C000-\$FFFFF during boot mode. The values in this register have no effect when the ZX-Uno is in run mode. 32 different values for this register allow addressing up to 512KB of SRAM. If ZX-Uno is expanded with more memory, more bits in this register will be used. The maximum manageable amount of memory is 4MB.

\$02 FLASHSPI

Direction	Value after user reset	Value after master reset	Value after poweron
Read/Write	Does not change	Does not change	00000000

Puerto de acceso al registro SPI conectado a la SPI Flash. Escribiendo un valor en este registro, se envía a la SPI Flash, si ésta está seleccionada. Leyendo un valor de este registro, se lee el último valor enviado por la SPI Flash, y además, la misma operación de lectura provoca que la SPI envíe un nuevo valor (que sería leído con la siguiente operación de lectura a este registro). Por esta razón, en operaciones de lectura de bloques, el primer byte leído con este puerto debe descartarse.

\$03 FLASHCS

Direction	Value after user reset	Value after master reset	Value after poweron
Read/Write	Does not change	Does not change	00000001

Only bit 0 is used. The value written to this register determines the status of the CS line of the SPI Flash (0 = Flash selected, 1 = Flash not selected).

\$04 SCancode

Direction	Value after user reset	Value after master reset	Value after poweron
Read/Write	Does not change	Does not change	Does not change

When reading, it allows to obtain the value of the last scancode generated by the keyboard. When writing, it allows to send commands to the keyboard.

\$05 KEYSTAT

Direction	Value after user reset	Value after master reset	Value after poweron
Read	Does not change	Does not change	Does not change

Several bits showing whether or not there is a new key pressed, or released, and whether this is an extended or normal key.

BSY	0	0	0	ERR	RLS	EXT	PEN
-----	---	---	---	-----	-----	-----	-----

- BSY: Uses 1 when a data transmission to the PS/2 port is still in progress. Wait for a value of 0 to start a new transmission.
- ERR: 1 when the last transmission to or from the PS/2 port had errors.
- RLS: 1 when the last event belongs to a key that has been released.
- EXT: 1 when the last event belongs to a key with extended code (E0+scancode).
- PEN: 1 when there is new data ready to be read in the SCANCODE register. After reading KEYSTAT, this bit is set to 0.

\$06 JOYCONF

Direction	Value after user reset	Value after master reset	Value after poweron
ReadWrite	Does not change	Does not change	00100001

Bits 0 to 2 show the operating mode of the keyboard mapped joystick (or of the second physical joystick if there is a splitter). Bits 4 to 6 for the operating mode of the physical joystick (DB-9 connector on the side). Bit 3 means autofire of the second joystick or keypad. Bit 7 means autofire of the main joystick. The values are: 000 = Disabled, 001 = Kempston, 010 = Sinclair 1, 011 = Sinclair 2, 100 = Protek/Cursor/AGF, 101 = Fuller, 110 = OPQAspM, 111 = reserved.

\$07 KEYMAP

Direction	Value after user reset	Value after master reset	Value after poweron
ReadWrite	Read later	Read later	Read later

On read, each access provides the next byte of the currently loaded ZX-Uno keymap. On write, the byte corresponding to the keymap marked by the current position is modified. In both cases, the address pointer is automatically incremented to point to the next byte of the keymap. This pointer returns to 0 automatically after a reset, a write to the \$FC3B register, or when the keymap is terminated. The keymap, in the current implementation, occupies 16384 bytes.

\$09 MOUSEDATA

Direction	Value after user reset	Value after master reset	Value after poweron
Read/Write	Does not change	Does not change	Does not change

Mouse PS/2 port data register. Used to read or send direct commands to the PS/2 mouse. For example: to initialise the mouse, the value \$F4 must be sent to this register.

\$0A MOUSESTATUS

Direction	Value after user reset	Value after master reset	Value after poweron
Read/Write	Does not change	Does not change	Does not change

PS/2 mouse port status register. The following bits are set:

BSY	0	0	0	ERR	0	0	PEN
-----	---	---	---	-----	---	---	-----

- BSY: set to 1 when a data transmission to the PS/2 port is still in progress. Wait for a value of 0 to start a new transmission.
- ERR: set to 1 when the last transmission to or from the PS/2 port had errors.
- PEN: set to 1 when new data is ready to be read from the MOUSEDATA register. After reading MOUSESTATUS, this bit is set to 0.

\$0B SCANBLCTRL

Direction	Value after user reset	Value after master reset	Value after poweron
Read/Write	Does not change	Does not change	00000000

Scandoubler control register and system speed. The following bits are defined:

TURBO	COPT	FREQ	FREQ	ENSCAN	VGA
-------	------	------	------	--------	-----

- TURBO: 00 to select 3.5 MHz, 01 to select 7 MHz, 10 to select 14 MHz and 11 to select 28 MHz. These bits are also updated with the value of bits D0-D3 of port \$8E3B, used in the ZX Prism to select the different speeds for the CPU.
- COPT: selects the way to generate the composite syncs for RGB and composite video: 0 to use the original Spectrum sync type. 1 to use sync pulses according to the PAL standard.
- FREQ: these three bits define the frequency of the master clock, from which all other clocks in the circuit are derived. Among others, the vertical refresh rate is also defined here, which can be used to improve compatibility with some VGA monitors that do not support a vertical refresh rate of 50Hz. The vertical refresh rate values are as follows:
 - 000 : 50Hz for 48K and Pentagon mode.
 - 001 : 50Hz for 128K mode
 - 010 : 52 Hz
 - 011 : 53 Hz
 - 100 : 55 Hz
 - 101 : 57 Hz
 - 110 : 59 Hz
 - 111 : 60 Hz
- ENSCAN: to 1 to enable the scanline effect in VGA mode. No effect if VGA mode is disabled.
- VGA: set to 1 to enable the scandoubler. The scandoubler output is the same as the normal RGB output, but with a doubling of the horizontal delay frequency. Set to 0 to use 15kHz RGB / composite video output.

\$0C RASTERLINE

Direction	Value after user reset	Value after master reset	Value after poweron
Read/Write	11111111	11111111	11111111

Stores the least significant 8 bits of the screen line where maskable interrupt should be triggered. A value of 0 for this register (with LINE8 also equal to 0) sets the raster interrupt to be triggered, if enabled, right at the start of the right edge of the line before the first screen line where the "paper" area begins. In other words: the line count of this interrupt assumes that a screen line is composed of: right edge + horizontal blanking interval + left edge + paper zone. If this assumption is made, the interrupt would be triggered at the beginning of the selected line. A value for RASTERLINE equal to 192 (with LINE8 equal to 0) triggers the raster interrupt at the beginning of the bottom edge. The line numbers for the end of the bottom edge and start of the top edge depend on the timings used. The highest value possible in practice for RASTERLINE corresponds to a raster interrupt triggered at the last line of the top edge (see RASTERCTRL).

\$0D RASTERCTRL

Direction	Value after user reset	Value after master reset	Value after poweron
Read/Write	00000001	00000001	00000001

Raster Interrupt Status and Control Register. The following bits are defined.

INT	0	0	0	0	0	DISVINT	ENARINT	LINE8
-----	---	---	---	---	---	---------	---------	-------

- INT: this bit is only available on read. It is set to 1 for 32 clock cycles from the time the raster interrupt is triggered. This bit is available even if the processor has interrupts disabled. It is not available if the ENARINT bit is set to 0.
- DISVINT: set to 1 to disable the vertical retrace maskable interrupts (the original ULA interrupts). After a reset, this bit is set to 0.
- ENARINT: set to 1 to enable raster line maskable interrupts. After a reset, this bit is set to 0.
- LINE8: saves bit 8 of the RASTERLINE value, so that any value between 0 and 511 can be set, although in practice, the largest value is limited by the number of lines generated by the ULA (311 in 48K mode, 310 in 128K mode, 319 in Pentagon mode). If a line number higher than the limit is set, the raster interrupt will not occur.

\$0E DEVCONTROL

Direction	Value after user reset	Value after master reset	Value after poweron
Read/Write	Does not change	00000000	00000000

Enable/disable register for different features. The following bits are defined.

DISD	ENMMU	DIROMSE L1F	DIROMSE L7F	DI1FFD	DI7FFD	DI7FFD	DITAY	DIAY
------	-------	----------------	----------------	--------	--------	--------	-------	------

- DISD: Set to 1 to disable the SPI hardware interface for SD (used in DivMMC and ZXMMC). Disabling this interface frees ports \$1F on write, \$3F, \$E7 and \$EB. After master reset, it is set to 0.
- ENMMU: set to 1 to enable the horizontal MMU used on the Timex Sinclair. Enabling this interface uses bit 7 of port \$FF, port \$F4 is used and a read to port \$FF returns the last value written to it. After master reset, it is 0.
- DIROM1F: the value of this bit is masked with the value of bit 2 of port \$1FFD according to the operation $\sim\text{DIROM1F} \& \$1FFD[2]$. The net result is that if this bit is set to 1, the system will work as if the value of bit 2 of port \$1FFD is always 0, regardless of the value written to it. After master reset, it is set to 0, which allows changes to bit 2 of \$1FFD to be taken into account.
- DIROM7F: the value of this bit is masked with the value of bit 4 of port \$7FFD according to the operation $\sim\text{DIROM7F} \& \$7FFD[4]$. The net result is that if this bit is set to 1, the system will work as if the value of bit 4 of port \$7FFD is always 0, regardless of the value written to it. After master reset, it is set to 0, which allows changes to bit 4 of \$7FFD to be taken into account.
- DI1FFD: set to 1 to disable the +2A/+3 compatible paging system. Disabling this interface frees the \$1FFD port on write. Note that the decoding of port \$7FFD, if active, is different depending on whether port \$1FFD is active or not.
- DI7FFD: to 1 to disable the 128K compatible paging system. Disabling this system disables the +2A/+3 paging system even if it is not explicitly disabled.
- DITAY: set to 1 to disable the second AY chip, thus disabling Turbo Sound mode.
- DIAY: set to 1 to disable the main AY chip. Disabling this chip disables the second AY chip, even if it is not explicitly disabled.

\$0F DEVCTRL2

Direction	Value after user reset	Value after master reset	Value after poweron
ReadWrite	Does not change	00000000	00000000

| Enable/disable register of different features (continuation of DEVCONTROL). The following bits are defined.

Resv	Resv	SPLITTER	DIMIXER	DISPECDRUM	DIRADES	DITIMEX	DIULAPLUS
------	------	----------	---------	------------	---------	---------	-----------

- Resv: This bit is reserved. In the current implementation, a 0 must be written to it if the register value is updated.
- SPLITTER: set to 1 to enable joystick splitter.
- DIMIXER: set to 1 to disable sound output.
- DISPECDRUM: set to 1 to disable SpecDrum/Covox support.
- DIRADES: set to 1 to disable radastanian mode. Note that if radastanian mode is not disabled, but ULaplus is disabled, when attempting to use radastanian mode, the datapath used in the ULA will not be as expected and the display behaviour in this case is not documented.
- DITIMEX: to 1 to disable Timex compatible display modes. Any writes to the \$FF port are therefore ignored. If the Timex MMU is enabled, a read to port \$FF will return 0.
- DIULAPLUS: set to 1 to disable the ULaplus. Any writes to the ULaplus ports are ignored. Reads to those ports return the value of the floating bus. Note however that the contention mechanism for this port still works even if it is disabled.

\$10 MEMREPORT

Direction	Value after user reset	Value after master reset	Value after poweron
ReadWrite	Does not change	00000000	00000000

\$40 RADASCTRL

Direction	Value after user reset	Value after master reset	Value after poweron
ReadWrite	00000000	00000000	00000000

Register for setting the radastanian mode and its characteristics. Writes to this register are ignored if the corresponding bit in DEVCTRL2 is set. The following bits are defined

Resv	EN1	EN0							
------	------	------	------	------	------	------	------	-----	-----

- Resv: This bit is reserved. In the current implementation, a 0 must be written to it if the register value is updated.
- EN1, EN0: Both bits must be set to 1 to enable Radastanian mode. If EN0 is enabled, but EN1 is disabled, the rest of the bits, from 2 to 7, are defined as implemented in the [ZEsarUX emulator](#) (see César Hernández's documentation on this).

\$41 RADASOFFSET

Direction	Value after user reset	Value after master reset	Value after poweron
ReadWrite	00000000	00000000	00000000

Contains the number of bytes to be added to the base address of the screen (4000h, 6000h, C000h or E000h depending on the configuration) to obtain the address where the first two pixels are located in Radastanian mode. It is a 14-bit register. To write a value, the 8 least significant bits are written first, followed immediately by the 8 most significant bits (the 2 most significant bits of this value are ignored). If the offset value is such that scanning the screen memory to create the image reaches the end of a 16KB page, scanning will continue at the beginning of that same page.

\$42 RADASPADDING

Direction	Value after user reset	Value after master reset	Value after poweron
ReadWrite	00000000	00000000	00000000

Contains the number of bytes - 64 that a scanline occupies in Radastanian mode. That is, if this register is 0, the length in bytes of a scanline is 64 bytes (128 pixels). If this register is 4, the length of a scanline is 68 bytes (136 pixels). If it is 255, the length of a scanline is $64+255=319$ bytes, or 638 pixels. If the offset value is such that scanning the screen memory to create the image reaches the end of a 16KB page, scanning will continue at the beginning of that same page.

\$43 RADASPALBANK

Direction	Value after user reset	Value after master reset	Value after poweron
ReadWrite	00000000	00000000	00000000

Register used to set which section of the ULAPlus palette will define the colours in Radastanian mode, and how the border will behave.

Resv	Resv	Resv	Resv	Resv	Resv	BOR3	RADPALQ UARTER
------	------	------	------	------	------	------	-------------------

- Resv: This bit is reserved. In the current implementation, a 0 must be written to it if the register value is updated.
- BOR3: Within the currently selected palette, indicates whether the border colour will be taken from entries 0 to 7 (0) or from entries 8 to 15 (1). It can be considered as bit 3 of the border colour in Radastanian mode.
- RADPALQUARTER: two bits indicating which section of the ULAPlus palette is to be used for the Radastanian mode. 00 for using the section of inputs 0 to 15. 01 for inputs 16 to 31, 10 for inputs 32 to 47 and 11 for using inputs 48 to 63.

\$80 HOFFS48K

Direction	Value after user reset	Value after master reset	Value after poweron
ReadWrite	Does not change	Does not change	\$38

Horizontal screen centering adjustment value for 48K ULA.

\$81 VOFFS48K

Direction	Value after user reset	Value after master reset	Value after poweron
ReadWrite	Does not change	Does not change	\$01

Vertical screen centering adjustment value for 48K ULA.

\$82 HOFFS128K

Direction	Value after user reset	Value after master reset	Value after poweron
ReadWrite	Does not change	Does not change	\$3A

Horizontal screen centering adjustment value for 128K ULA.

\$83 VOFFS128K

Direction	Value after user reset	Value after master reset	Value after poweron
Read/Write	Does not change	Does not change	\$01

Vertical screen centering adjustment value for 128K ULA.

\$84

Direction	Value after user reset	Value after master reset	Value after poweron
HOFFSPEN	Read/Write	Does not change	Does not change

Horizontal screen centering adjustment value for Pentagon ULA.

\$85 VOFFSPEN

Direction	Value after user reset	Value after master reset	Value after poweron
Read/Write	Does not change	Does not change	\$00

Vertical screen centering adjustment value for Pentagon ULA.

\$A0 DMACTRL

Direction	Value after user reset	Value after master reset	Value after poweron
Read/Write	Does not change	00000000	00000000

\$A1 DMASRC

Direction	Value after user reset	Value after master reset	Value after poweron
Read/Write	Does not change	00000000	00000000

\$A2 DMADST

Direction	Value after user reset	Value after master reset	Value after poweron
Read/Write	Does not change	00000000	00000000

\$A3 DMAPRE

Direction	Value after user reset	Value after master reset	Value after poweron
Read/Write	Does not change	00000000	00000000

\$A4 DMALEN

Direction	Value after user reset	Value after master reset	Value after poweron
Read/Write	Does not change	00000000	00000000

\$A5 DMAPROB

Direction	Value after user reset	Value after master reset	Value after poweron
Read/Write	Does not change	00000000	00000000

\$A6 DMASTAT

Direction	Value after user reset	Value after master reset	Value after poweron
Read/Write	Does not change	00000000	00000000

\$C6 UARTDATA

Direction	Value after user reset	Value after master reset	Value after poweron
Read/Write	Does not change	00000000	00000000

\$C7 UARTSTAT

Direction	Value after user reset	Value after master reset	Value after poweron
Read/Write	Does not change	00000000	00000000

\$C8 - \$DF RESERVED

Direction	Value after user reset	Value after master reset	Value after poweron
Read/Write	Definido por el usuario	Definido por el usuario	Definido por el usuario

ZXUNO registers reserved for experiments or private use.

\$F0 SRAMADDR

Direction	Value after user reset	Value after master reset	Value after poweron
Read/Write	Does not change	00000000	00000000

\$F1 MADDRINC

Direction	Value after user reset	Value after master reset	Value after poweron
Read/Write	Does not change	00000000	00000000 ===== \$F2 SRAMDATA [align="center",options="header"]

| Direction | Value after user reset | Value after master reset | Value after poweron
| Lectura/Escritura | Does not change | 00000000 | 00000000

\$F3 VDECKCTRL

Direction	Value after user reset	Value after master reset	Value after poweron
Read/Write	Does not change	00000000	00000000

\$F7 AUDIOMIX

Direction	Value after user reset	Value after master reset	Value after poweron
Read/Write	Does not change	Does not change	10011111

Control of the left/right channel mix from the 4 channels of the AY-8912. The following groups of 2 bits each are defined:

---	CHANNELA	CHANNELB	CHANNELC	BEEPDRUM
-----	----------	----------	----------	----------

- BEEPDRUM : Channel for Beeper and Specdrum.
- The meaning of each group of 2 bits is: 00 = Mute, 01 = Right channel, 10 = Left channel, 11 = Both.

\$FB AD724

Direction	Value after user reset	Value after master reset	Value after poweron
ReadWrite	Does not change	Does not change	00000000

Control of the operating mode of the AD724 encoder chip. The following bits are defined:

Resv	MODE								
------	------	------	------	------	------	------	------	------	------

- Resv : the meaning of these bits is reserved and must not be altered.
- MODE : operating mode of the AD724. 0 = encodes PAL standard. 1 = encodes NTSC standard.

\$FC COREADDR

Direction	Value after user reset	Value after master reset	Value after poweron
ReadWrite	Does not change	Does not change	\$058000

Stores the address, within SPI memory, of the start of the core to be booted. To store a different address, three writes must be made to this register, containing the three bytes of the address, in order from most significant to least significant. On read, each access to this register returns a part of the last stored address, from the most significant part to the least significant part.

\$FD COREBOOT

Direction	Value after user reset	Value after master reset	Value after poweron
Read	Does not change	Does not change	Does not change

Boot control register. Writing a 1 to bit 0 of this register (all other bits are reserved and must remain at 0) triggers the FPGA's internal mechanism to start another core. The start address of this second core will be the last one written using the COREADDR register.

\$FE SCRATCH

Direction	Value after user reset	Value after master reset	Value after poweron
ReadWrite	Does not change	00000000	00000000

\$FF COREID

Direction	Value after user reset	Value after master reset	Value after poweron
Read	Read later	Read later	Read later

Each read operation provides the next ASCII character of the string containing the current revision of the ZX-Uno core. When the string ends, subsequent reads output bytes with the value 0 (at least on) until the string starts again. This pointer returns to 0 automatically after a reset or a write to the **\$FC3B** register. The delivered characters that are part of the string are standard printable ASCII (codes [32-127](#)). Any other value indicates that this register is not operational.

Links

[ZX-Uno](#)

[foroFPGA](#)

[canal oficial de Telegram de ZX-Uno](#)

[ZX-Uno FAQ](#)

[Repositorio de ZXUno4All](#)

[ZXUnCore en foros va de Retro](#)

[ZXUno4All en foros va de Retro](#)

[Repositorio de ZXUnoPS2](#)

[Repositorio de joy2ps2](#)

[Guía rápida del ZX-Uno](#)

[Wiki de ZX-Uno.](#)

[Core ZX Spectrum](#)

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

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

[Configuración de teclado de ZX-Uno.](#)

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

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

[ESXDOS Manual](#)

[WiFi \(RetroWiki\)](#)

[Cargando Leches 2.0](#)

[WiFi en ZX-Uno](#)

[Core de ZX-Uno Test UART \(WiFi\)](#)

[Network tools for ZX-Uno pack](#)

[ESP8266 AT Instruction Set](#)

[Vídeos Radastanianos](#)

[Nuevo core zx48](#)

[ZX 48 for ZX-UNO \(Kyp\)](#)

[Core MSX](#)

[MSX1FPGA](#)

[MSX Pack](#)

[Nextor para MSX](#)

[Nextor User Manual](#)

[MSX-DOS](#)

[Atom Software Archive en carpeta ATOM](#)

[Teclado Core Atom](#)

[Core de NES para ZX-Uno](#)

[ColecoFPGA on GitHub](#)

[TOSEC: Magnavox Odyssey 2 \(2012-04-23\)](#)

[Videopac G7000 / Odyssey2 for FPGA](#)

[zx123_tool](#)

[esxdos](#)

[ZX-Modules.](#)

[Long Filename Browser de Bob Fossil](#)

[New NMI Handler de Dr. Slump](#)

[Fuzix](#)

[SmartROM](#)

[ZX Tape Player](#)

[PlayZX](#)

[Tapir](#)

[Fuse Utilities](#)

[MakeTSX](#)

[Lynx2Wav](#)

[2lynx2wav](#)

[cgc2wav](#)

[mzf2wav](#)

[RetroConverter](#)

[Lince Script](#)

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

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

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

[ZX-Uno Secret ROM](#)