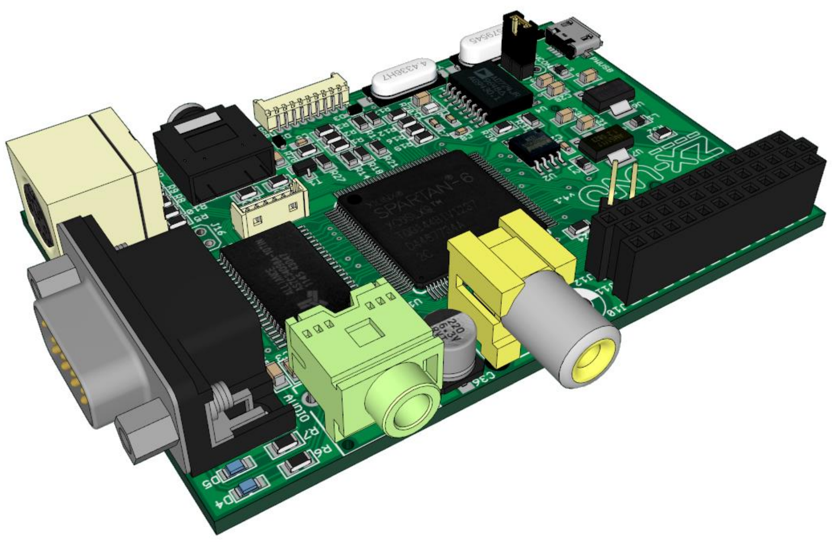
ZX-UNO

CLON ORDENADOR ZX-SPECTRUM BASADO EN FPGA

Superfo · AVillena · McLeod · Quest · Hark0

MANUAL



Author: Manu (April 2016)
Last major revision: Uto (July 2016)
Last minor revision: 1024MAK (September 2016)

Send your feedback to @uto_dev o utodev en gmail.com

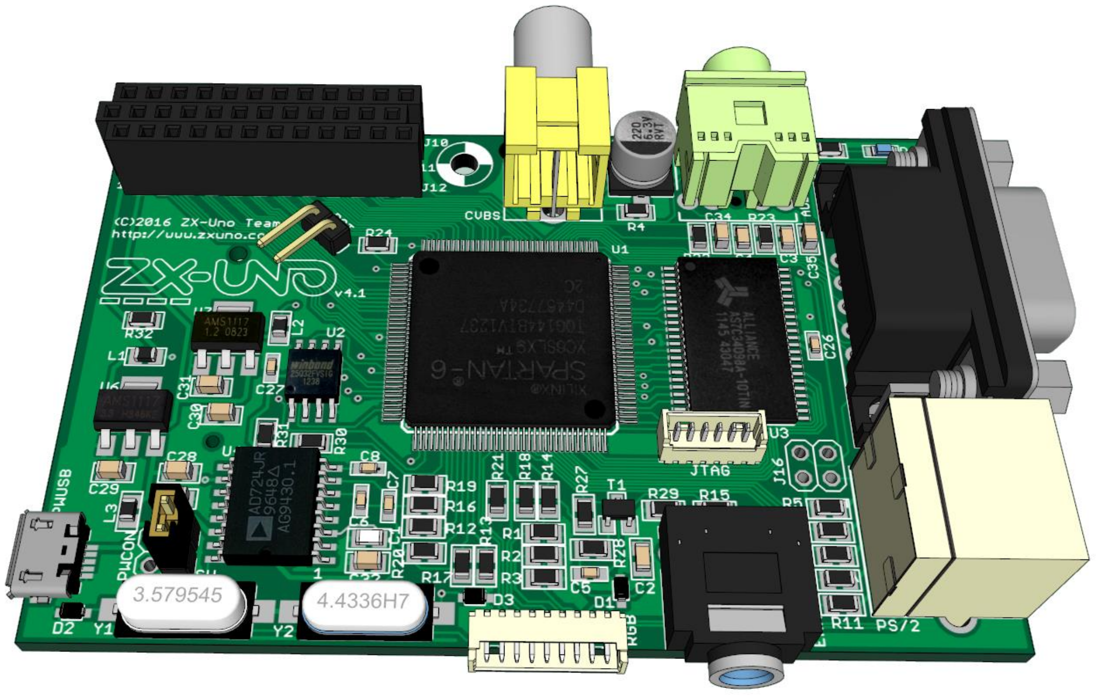
Contents

Part I - Introduction

What is the ZX-Uno?

The ZX-Uno is a board that allows several computers to be implemented, mirroring the original hardware as close as possible. That is made possible by using an FPGA, that may be set up thanks to circuit definition languages such as VHDL or Verilog.

Although it was originally designed to clone Sinclair's ZX Spectrum and probably that computer implementation is the most complete one, ZX-Uno may also implement some other computers and consoles (although due to space requirements, only 9 cores at once can be loaded). Those alternative cores are Sam Coupe, Jupiter Ace, Apple II, Acorn Atom, BBC Micro, Acorn Electron, Oric Atmos and VIC-20, together with implementations for Master System, NES and Atari 2600 consoles. This manual is mainly dedicated to the ZX Spectrum core, there is a section dedicated to other cores though.



The ZX-Uno board is designed with same form factor than the original Raspberry Pi model A, having most of its connections placed at same place, allowing us to use boxes designed for that Raspberry Pi with minimal modifications.

The ZX-Uno FPGA is a Spartan-6, with features that can target mainly 8 bits consoles and computers.

Fast setup

What should I do to just boot up my ZX-Uno and play some games?

Simplest setup for ZX-Uno requires:

1) A TV or display unit with composite video and audio connectors. If your TV doesn't have one of those but has a SCART connection there are adapters allowing you to switch from one to another.

2) A microUSB power supply, just like those used for Android phones (with microUSB connector). Also, is often possible to power it from your TVs USB connectors, if available.

MicroUSB charger

3) An SD card, formatted with FAT format, and including some files (see below).

SD card

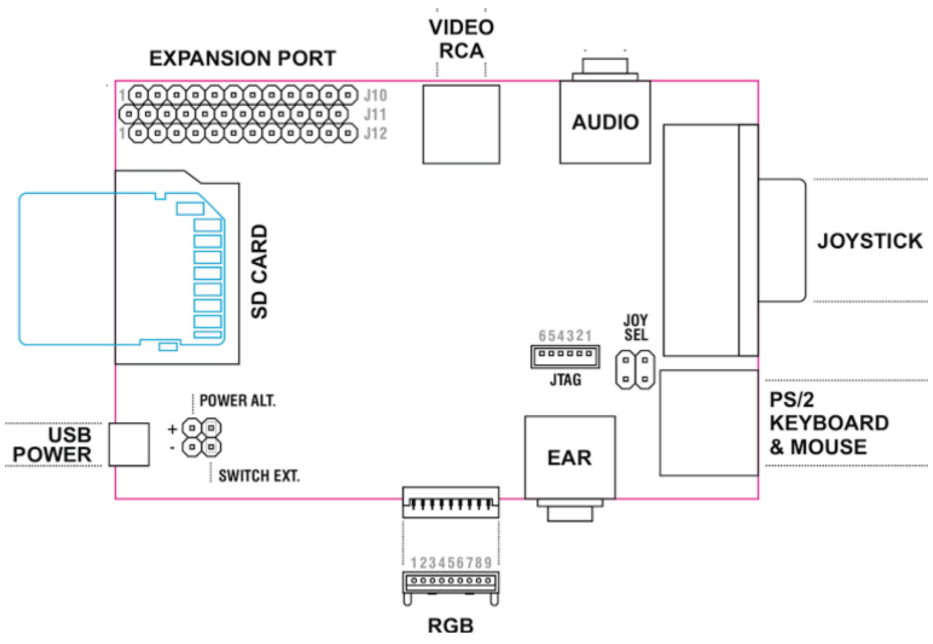
4) A PS/2 keyboard, or a USB keyboard that is PS/2 compatible and a USB to PS/2 adapter. Please notice most USB keyboards are not compatible with PS/2 so they won't work even with an adapter.

PS/2 Keyboard

5) Cables for audio and video connections:

Composite video and audio cables

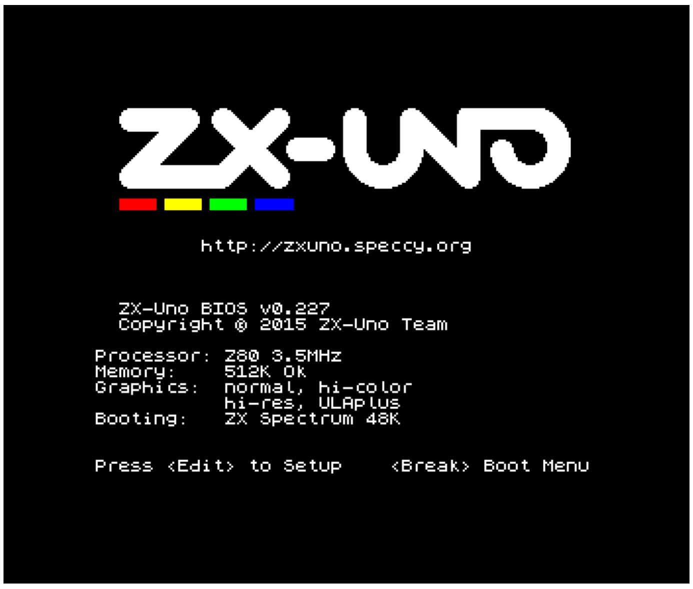
To set up your ZX-Uno, we will follow the following steps, checking this diagram:



Board connections

1) Connect the RCA (phono) connection (the yellow one) to your TV's composite video connector (yellow).
2) Connect the audio Jack to the audio connection on your ZX-Uno, and to red and white RCA connections (audio right and audio left) on your TV
3) Plug keyboard into keyboard connection
4) Finally, connect power

Once the power is connected, the ZX-Uno should boot and show the ZX-Uno boot screen, followed by the classic © 1982 Sinclair Research Ltd message, but we cannot load any game.



```
http://zxuno.speccy.org

 ZX-Uno BIOS v0.227
 Copyright © 2015 ZX-Uno Team

Processor: Z80 3.5MHz
Memory:    512K Ok
Graphics:  normal, hi-color
           hi-res, ULAplus
Booting:   ZX Spectrum 48K

Press <Edit> to Setup    <Break> Boot Menu
```

ZX-Uno boot screen

Note: it is possible that your ZX-Uno does not include the Sinclair ROM, but the SE Basic IV Anya ROM (http://cheveron.github.io/sebasic4). If that's the case we will see later how to load the ROMs you want into your ZX-Uno.

Preparing the SD card

We can already use a cassette deck to load games, using an original cable, or use ESXDOS to be able to load from the SD card. To do that we should format the card with a FAT format, and include these folders in the card's root folder. Also add some games/utilities in the card in the tape (TAP), snapshot (SNA, Z80) or disk (TRD) formats. For the time being ESXDOS is only able to show 8 characters per file name, so try not to save files with long names (rename them), that is, instead of jetsetwilly.tap, rename to jsetwill.tap or similar.

If the link above doesn't work or you are reading a printed version of this manual, this is the link where those folders are:

http://guest:zxuno@www.atc.us.es/svn/zxuno/software/esxdos/0.8.6-beta4/

Loading games

Once the card is set up, we insert it into the ZX-Uno and perform a reset (Ctrl+Alt+Del). This time, after the ZX-Uno boot screen we should see the ESXDOS boot screen, where a message saying our card has been detected will be shown. If not, please check previous steps, or try another SD card.

Once ZX-Spectrum boots, we can can force a NMI (Ctrl+Alt+F5) and a menu will be shown allowing us to choose a game/utility using cursor keys, and load them pressing ENTER..

Note: you can extract and insert the SD card while ZX-Uno is on, but is not recommended to do it while ZX-Uno is writing on it. Doing so may cause data loss.

Compatibility issues

There had been some problems before with some games that were activating paging by mistake (some old Ultimate games mainly) when the ZX-Uno was started (booted) in 128K mode. Now when the ZX-Uno boots with a 48K ROM, it resembles a pure 48K machine, so that the problem no longer occurs. Should you need to boot in 128K mode and load these games, you can do an OUT 32765, 48 to avoid those problems.

On the other hand, some 128K games that assume a valid value at the BANKM system variable don't work if DivMMC is active, as ESXDOS forces USR 0 mode in 128K computers. In those cases a POKE 23388,16 before loading them is used to fix the problem.

You can use those POKEs and OUT or just load these tap files before, so you don't have to remember the ports/addresses and values.

- Tap to deactivate paging
- Tap to insert a valid value in BANKM

If you are reading a printed version of this manual, in this link there is a tutorial about using DivMMC that includes links to those two tap files. The tutorial is in Spanish, but you should be able to locate the links for those tap files easily:

http://bit.ly/2azqTWO

If despite of that, the game still doesn’t work, maybe you have found some other incompatibility with the new features in ZX-Uno, as ULAplus, Timex modes, etc.

You can try to disable those features and see if it then works okay. You can do it like this:

1) Download tap file for ZXUC from https://github.com/Utodev/ZXCU

2) Run it, enter hardware options, and disable ULAPlus, Radastan mode, Timex modes and MMU Timex. Press B to main menu, and E to exit.

3) Load the game without resetting first.

If it still doesn’t work, please report in the ZX-Uno forums.

Part II – Technical Guide

Connections and peripherals

The ZX-Uno has several connections available, for power input, video output, audio input and output, keyboard, joystick, etc.

To power up the ZX-Uno we can use any microUSB power unit that supplies 5V DC, for example a power supply from an Android mobile telephone, or even use the TV USB Port.

TV or display unit

Not all the cores implemented by ZX-Uno support all connection methods, but these are the methods available on the motherboard:

- **Composite video:** The yellow RCA (phono) connector is the easiest way to connect a ZX-Uno to a TV. Some TVs don't have composite input, but if they have a SCART connection and adapter will do the trick.
- **RGB**: Provides a clearer, better quality signal and works in all cores supporting composite video. A custom cable is needed though, you can find the pin-out of that connector in the appendix of this manual.
- **VGA**: using the RGB connector, you can connect to a VGA display unit. Not all cores support VGA output though.

Sound

You can use a Y cable like the one shown in the introduction chapter, or you can use an adapter like the one below.

You can also skip using TV for sound and just use standard PC speakers and plug them into the audio connection.

Keyboard

Keyboards for the ZX-Uno must be PS/2 Keyboards, or a dual protocol PS/2+USB keyboard. For those dual protocol keyboards an adapter should be used. You should take into account that not every USB keyboards is dual, and thus many of them won't work even with an adapter. There are some keyboards that have proven to be compatible with ZX-Uno despite being USB keyboards, see this forum thread.

These are the most important key combinations to be used in ZX-Uno boot:

Caps Lock (Caps Shift+2, Caps Lock)	Choose core at boot time
Esc (Caps + Space, Break)	Choose Spectrum ROM at boot time
F2 (Caps +1, Edit)	Enter the BIOS

Also, once the ZX-Uno is running, these are useful keyboard combinations:

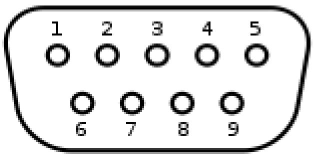
Ctrl	Symbol Shift
Left Windows key	Caps Shift
Control+Alt+Del	Reset
Control+Alt+Backspace	Master Reset
Control+Alt+F5	NMI (if you have ESXDOS active it will show the card contents, but other ROMS may have other effects)
Scroll Lock	Changes the output mode of the RGB connection, so you can try different modes until your display unit shows ZX-Uno screen.

Joystick

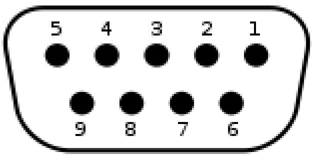
The **ZX-Uno** is designed to use controller pads and joysticks using the Atari specifications, that is, those that originally worked with an Atari VCS console. That protocol was later adopted by machines like the ZX-Spectrum (Kempston), Amstrad CPC, MSX, CBM 64 and Master System. Other joysticks such as those from Amstrad CPC plus or MSX can cause problems.

Those joysticks using the SJS-1 protocol (those created with the Spectrum +2) are not compatible and require an adapter.

This table shows the pinout of different systems:



ZX-Uno



Joystick

Pin	ZX-Uno	Atari VCS	Master System	Mega Drive	SJS-1
1	Up	Up	Up	Up	-
2	Down	Down	Down	Down	GND
3	Left	Left	Left	Left	-
4	Right	Right	Right	Right	Up
5	Customizable	Paddle B	+5V	+5V	Button 1
6	Button 1	Button 1	Button 1	/A, /B	Right
7	Customizable	+5V	-	Select OUT	Left
8	GND	GND	GND	GND	GND
9	Button 2	Paddle A	Button 2	/Start, /C	Down

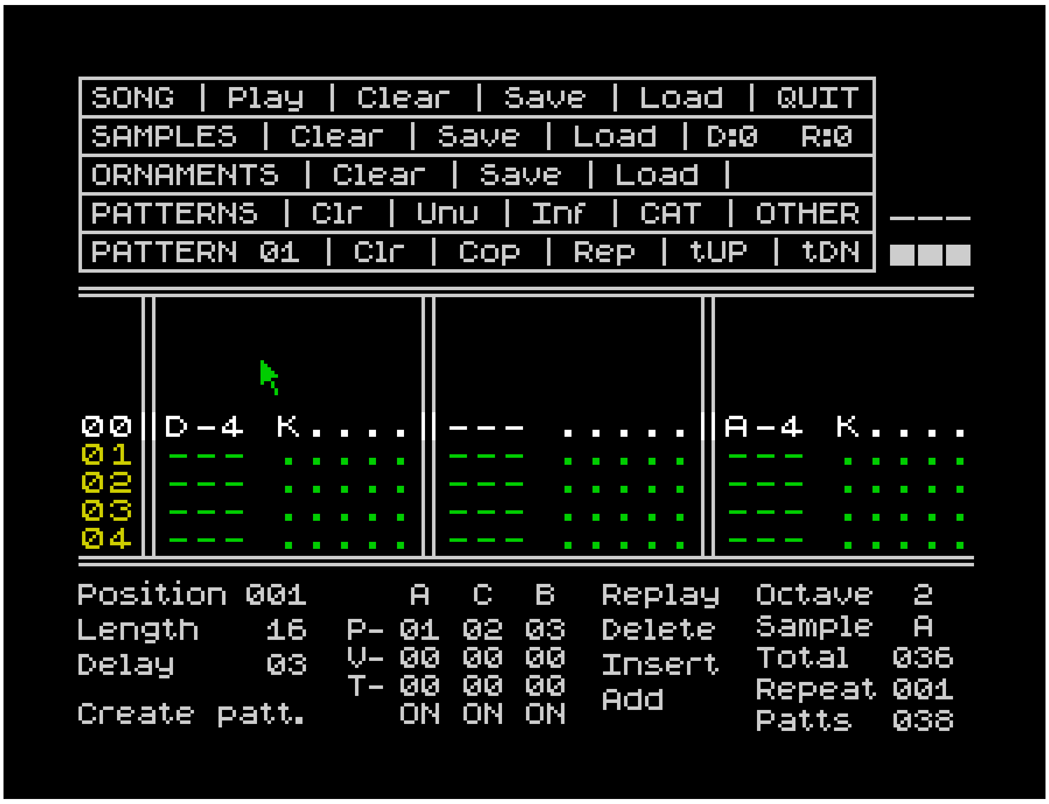
The ZX Spectrum core in ZX-Uno allows the Atari Joystick to work as a Kempston, Cursor or Sinclair 1 or 2 joystick. You can use the unofficial ESXDOS command **.JOYCONF** to choose mode, choose it from the BIOS or use ZXUC application (https://github.com/Utodev/ZXCU)

Other than the classic joystick plugged into the joystick port, the ZX-Uno simulates another joystick using the numeric keyboard cursor keys, and the ALT key as fire button. You can choose which joystick type (Kempston, Cursor, etc.) this virtual keyboard emulates with ZXUC, .JOYCONF and BIOS too.

Mouse

Despite the ZX-Uno having just one PS/2 connection, a Y cable like thise, can be used. Note the Startech model linked here has had the colours changed, so plug green in purple and purple in green.

If you connect a mouse to your Zx-Uno, it emulates a *Kempston Mouse*, so you can use it with compatible software such as graphic design software, music trackers or even some patched games as “*Lemmings*”, “*Operation Wolf*”, “*Arkanoid*” or “*R-Type*”.



```
SONG | Play | Clear | Save | Load | QUIT
SAMPLES | Clear | Save | Load | D:0   R:0
ORNAMENTS | Clear | Save | Load |
PATTERNS | Clr | Unu | Inf | CAT | OTHER   ---
PATTERN 01 | Clr | Cop | Rep | tUP | tDN   ■■■

00 || D-4  K..... || ---  ...... || A-4  K.....
01 || ---  ...... || ---  ...... || ---  ......
02 || ---  ...... || ---  ...... || ---  ......
03 || ---  ...... || ---  ...... || ---  ......
04 || ---  ...... || ---  ...... || ---  ......

Position 001       A  C  B   Replay   Octave  2
Length    16    P- 01 02 03  Delete   Sample  A
Delay     03    V- 00 00 00  Insert   Total   036
                T- 00 00 00  Add      Repeat  001
Create patt.       ON ON ON           Patts   038
```

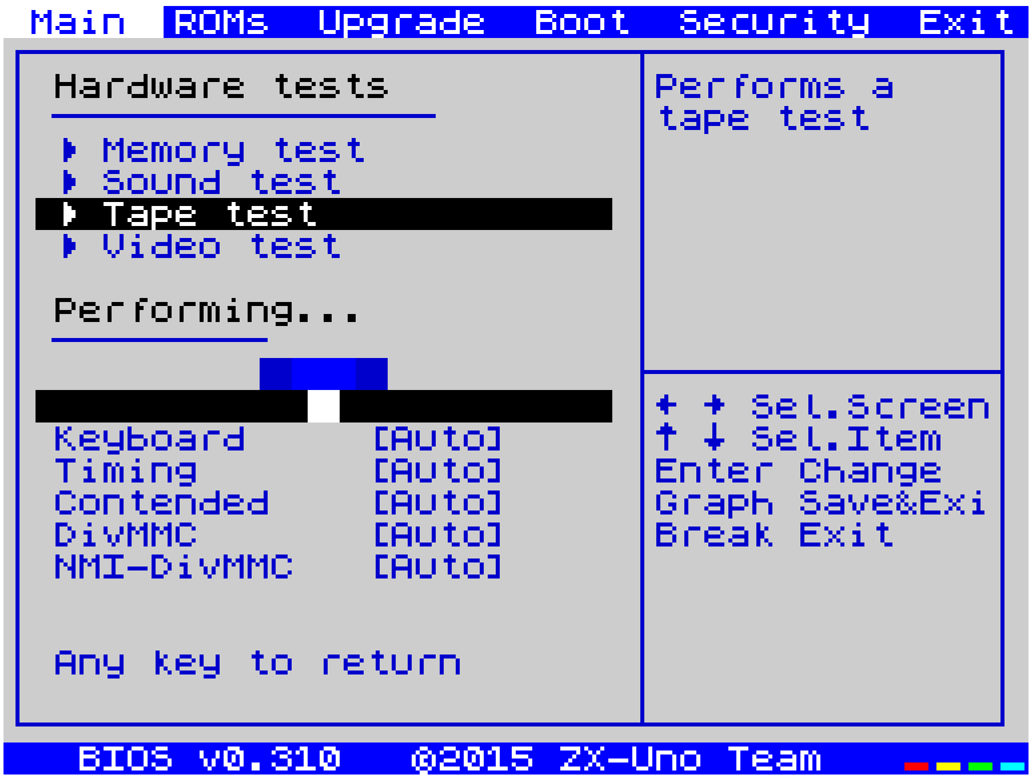
Loading from audio source

Despite the ZX-Uno being ready to quickly load using ESXDOS, you can load software from tape from cassette deck, or use players designed for computers or mobile telephones to load TAP, TZX or PZX files.

- PC: Tapir
- Android: TeeZiX, tapDancer
- Speccy Tape

In any of these cases you only need to plug the player into your ZX-Uno using the EAR connection (yes, old style).

To make sure the volume is loud enough (some music players, Android devices and computers have very low volume) there is a tape test available in the BIOS. If the volume is not loud enough, you can use a portable audio amplifier.



Hardware tests

- Memory test
- Sound test
- Tape test
- Video test

Performing...

```
Keyboard      [Auto]
Timing        [Auto]
Contended     [Auto]
DivMMC        [Auto]
NMI-DivMMC    [Auto]
```

Any key to return

Performs a tape test

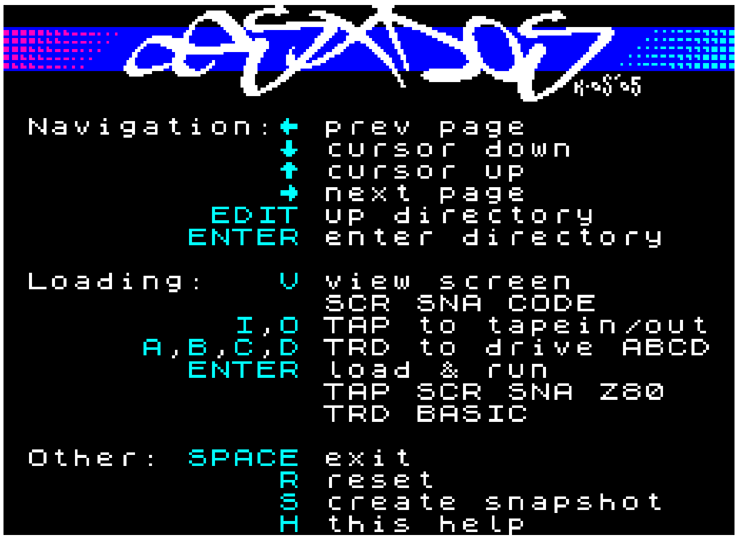
```
← → Sel.Screen
↑ ↓ Sel.Item
Enter Change
Graph Save&Exi
Break Exit
```

BIOS v0.310 @2015 ZX-Uno Team

Using ESXDOS

ESXDOS is an advanced firmware for divIDE and divMMC, that among other features allows extended commands and loading tape images and snapshot directly from SD card within milliseconds. If we have chosen a boot ROM supporting ESXDOS we can access and browse the menu using NMI (Control+Alt+F5).

Once in the NMI listing, you can have several options, press H to see help:



```
Navigation: ←     prev page
            ↓     cursor down
            ↑     cursor up
            →     next page
         EDIT     up directory
        ENTER     enter directory

Loading:    V     view screen
                  SCR SNA CODE
          I,O     TAP to tapein/out
      A,B,C,D     TRD to drive ABCD
        ENTER     load & run
                  TAP SCR SNA Z80
                  TRD BASIC

Other:  SPACE     exit
            R     reset
            S     create snapshot
            H     this help
```

Keyboard shortcuts in the file selector

Other than that, ESXDOS includes new commands than can be used from BASIC, preceding them by a dot, thus they are named dot commands.

For instance, one of those commands is .LS, that lists contents of current folder:

```
.LS
```

To see the list of available commands type:

```
.LS BIN
```

Which will then list the contents of the BIN folder, containing the dot commands. Most dot commands resemble UNIX commands (CD, LS, MV, MKDIR, etc.), and some others have a more or less intuitive function (.DUMPMEM, .PLAYWAV,...). Finally, there are a few commands specific to ZX-Uno. Most of them will show you how to use them if you add the –h parameter:

```
.ZXUNOCFG -h
```

Those specific to ZX-Uno are:

- Change to VGA mode:
  - **.ZXUNOCFG -v1** (no scanlines) **/ .ZXUNOCFG -v2** (scanlines)

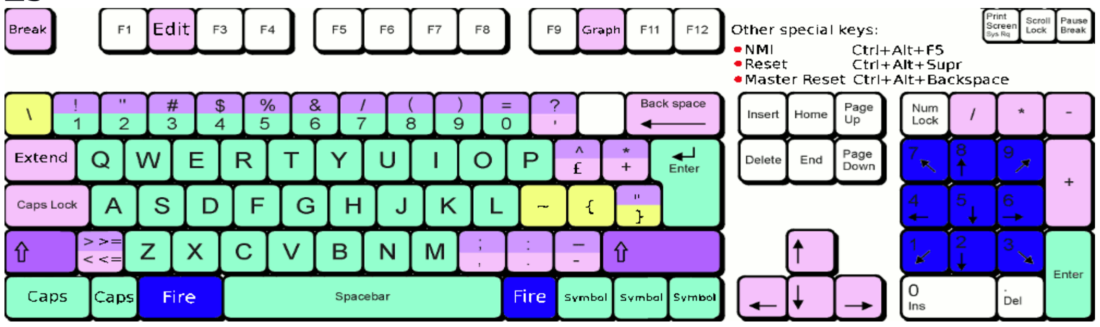
- Joysticks setup
  - **.JOYCONF**

- Boot core #n
  - **.CORE n**

- Change Keyboard layout
  - **.KEYMAP layout**

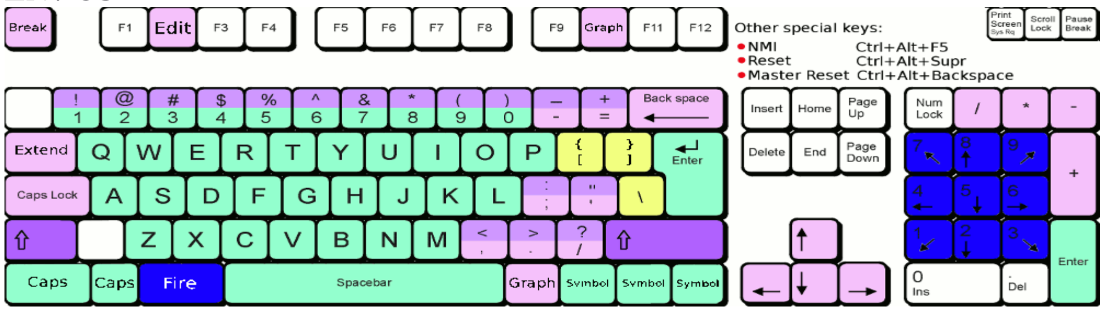
Keyboard layouts should be in the SYS/keymaps folder in the SD card. The ZX-Uno defaults to Spanish keyboard layout, but this can be easily changed from BIOS. The new keyboard layout will then be remembered.

ES



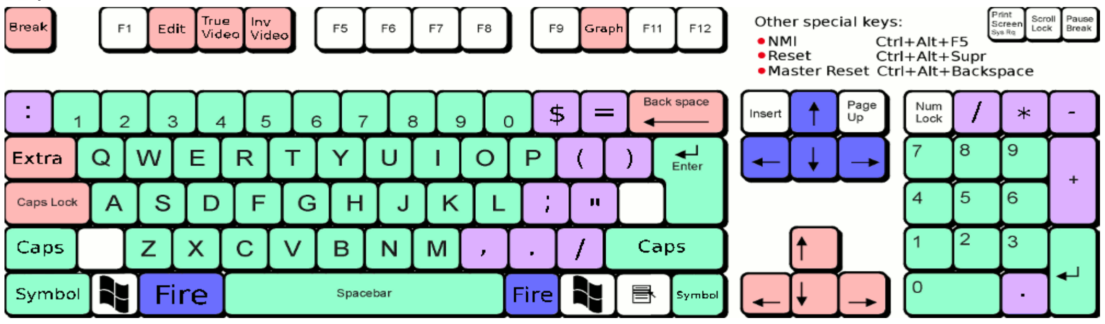
Other special keys:
- NMI Ctrl+Alt+F5
- Reset Ctrl+Alt+Supr
- Master Reset Ctrl+Alt+Backspace

EN / US



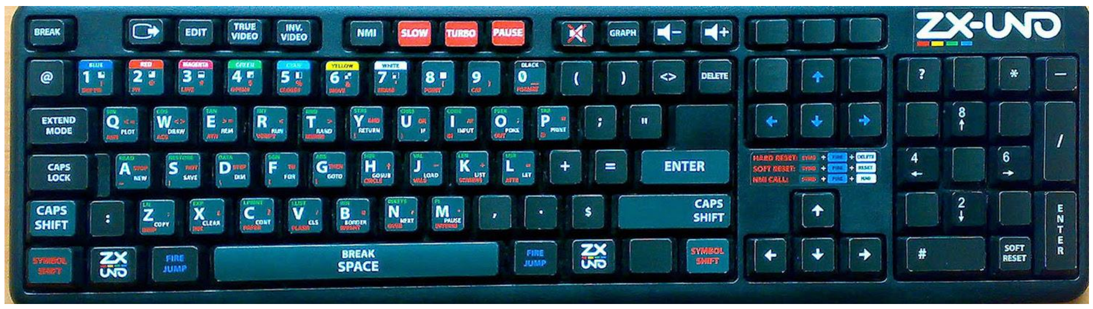
Other special keys:
- NMI Ctrl+Alt+F5
- Reset Ctrl+Alt+Supr
- Master Reset Ctrl+Alt+Backspace

AV



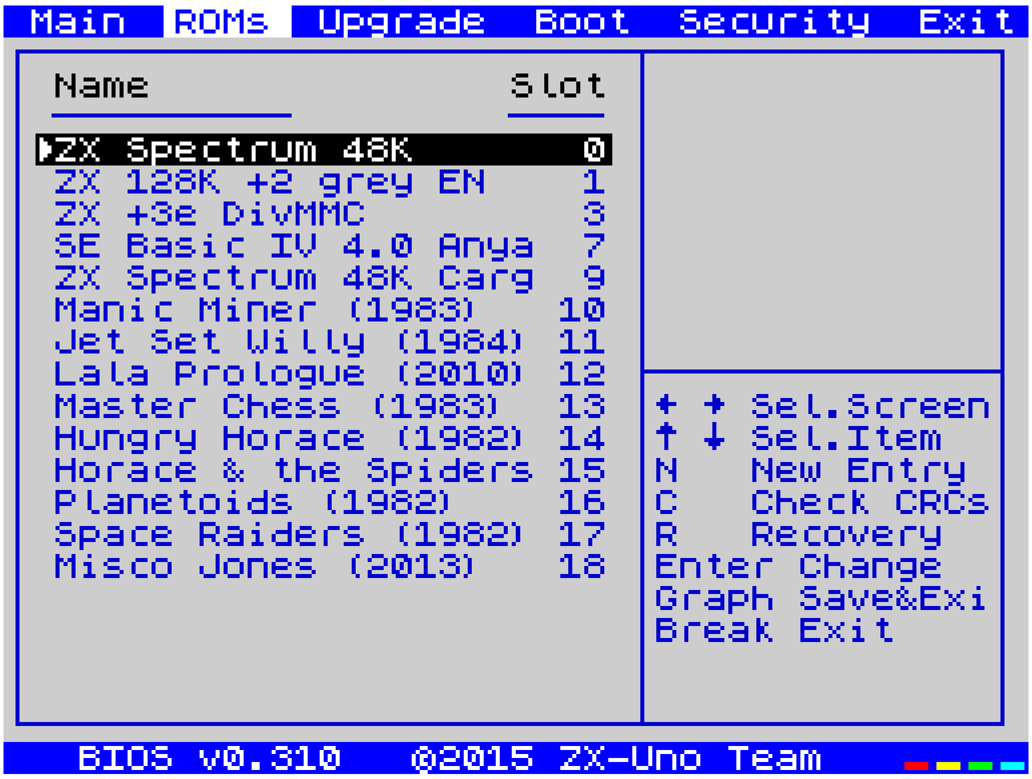
Other special keys:
- NMI Ctrl+Alt+F5
- Reset Ctrl+Alt+Supr
- Master Reset Ctrl+Alt+Backspace

MJ



Change default boot Spectrum ROM

The ZX Spectrum core allows any compatible ROM to be used (48K ROMs, 128K ROMs and even Interface 2 games, the original ones and those created later)



```
 Main  ROMs  Upgrade  Boot  Security  Exit

 Name                     Slot

▶ZX Spectrum 48K             0      ← → Sel.Screen
 ZX 128K +2 grey EN          1      ↑ ↓ Sel.Item
 ZX +3e DivMMC               3      N   New Entry
 SE Basic IV 4.0 Anya        7      C   Check CRCs
 ZX Spectrum 48K Carg        9      R   Recovery
 Manic Miner (1983)         10      Enter Change
 Jet Set Willy (1984)       11      Graph Save&Exi
 Lala Prologue (2010)       12      Break Exit
 Master Chess (1983)        13
 Hungry Horace (1982)       14
 Horace & the Spiders       15
 Planetoids (1982)          16
 Space Raiders (1982)       17
 Misco Jones (2013)         18

   BIOS v0.310    @2015 ZX-Uno Team
```

The Boot ROM can be chosen by pressing ESC (break) when the ZX-Uno shows its loading screen, this will affect just that boot. If you want to change boot ROM permanently you will need to enter the BIOS, go to the ROMS menu, and press enter on the chosen one, then select “Set active”.

If you have already booted your ZX-Uno remember that in order to see the boot screen again you have to power off/on the ZX-Uno or perform a hard reset (Ctrl+Alt+Backspace).

Alternative to ESXDOS : +3E

ESXDOS is designed to read SD cards using the FAT format, but before ESXDOS was developted, +3E existed, being an extension for Spectrum +3DOS, able to read IDE devices like hard disk drives and CF cards. If you want your ZX-Uno to work with +3E you need to create an IDEDOS partition. To do so, you can dedicate a SD card for it and format it from BASIC, or follow the official instructions to save an extended partition, and be able to use the rest of the card for FAT and ESXDOS.

If we use a dedicated card, you can format it like this:

```
FORMAT TO 0, nn
```

Where nn depends on card size and the number of partitions we want (max is 16). For instance, for a 256Mb partition value for nn is 15.

Then we create a data partition and we mount it permanently:

```
NEW DATA “Tests”, 16
MOVE “C:” IN “Tests” ASN
LOAD “C:” ASN
```

In this case, the data partition wold be *Tests*, size 16MB, and mapped always as C: drive as we have added the ASN parameter.

Is highly recommended reading the +3E command reference and the +3E documentation to get all the features that this ROM can provide.

Part II – Developing for the ZX-Uno

Although there is documentation enough for most extra features provided by ZX-Uno, with the exception of Radastan video mode, this manual will cover those features provided by ZX-Uno that were not in the original 48K Spectrum – even when some of them were in the Timex models or 128K models.

Detecting the ZX-Uno

In case you are going to use a ZX-Uno feature it's always better trying to detect that feature specifically, so your software has a chance of running in other machines supporting the same feature. In case that is not possible you may not be able to detect the feature directly, and so you may need to detect the ZX-Uno itself.

To detect a ZX-Uno, a better option is to access the COREID register. To do that, you need to know that this register returns an ASCII string, one char per reading, and never returns values lower than ASCII 32 or greater than ASCII 127. If an out of range value is returned, or an empty string is returned, then it's not a ZX-Uno.

This code checks for a ZX-Uno:

```
10 OUT 64571, 255
20 LET A$=””
30 LET A= IN 64827
40 IF (A<32) OR (A>127) THEN GOTO 60
50 IF (A<>0) THEN LET A$ = A$ +CHR(A): GOTO 30
60 IF A$=”” THEN GOTO 80
70 PRINT “IT'S A ZX-UNO”: STOP
80 PRINT “IT'S NOT A ZXUNO”
```

Graphic modes and colours

The ZX-Uno supports four different graphic modes:

1) Standard ZX Spectrum mode: 256x192 pixels, with a paper/ink/bright/flash attribute per each 8x8 pixels block.
2) Timex HiColour mode, 256x192 pixels, with a paper/ink/bright/flash attribute per each 8x1 pixels block.
3) Timex HiRes mode, 512x192 pixels, with just one INK and one PAPER colour for the whole screen. No colour clash (as it's monochrome).
4) Radastan mode, 128x96 pixels, linear, 16 colours per pixel, no colour clash.

Also, and thanks to ULAplus, it is possible to change colour palette, and get up to 64 colours on screen at the same time.

The ZX-Uno will boot using the standard ZX Spectrum mode, but we won't talk about it as it's well known. About the other modes, you need to take into account that Timex computers allow having two memory banks for the screen, that can be switched when needed. First bank is located at the usual memory address (0x4000 or 16384) and is named Screen 0, while the second one is located at 0x6000 (24576) and is named Screen 1. When using the standard mode we can use that feature of switching banks to change the content on the screen fast, having a second screen ready at 0x6000 and commanding ZX-Uno to change to 0x6000, then we can prepare another screen at 0x4000 and switch back to 0x4000, etc. We will see later how to switch from Screen 0 to Screen 1 and vice-versa.

HiColour mode uses the standard screen memory (Screen 0) to define the pixels in the screen, using the usual 6144 bytes to do that, but the attributes, instead of being located right after the pixels, to complete the usual 6912 bytes, are located in the Screen 1 zone. Attributes are placed exactly like the screen pixels are defined, first row first, then 8th row, etc.

HiRes mode uses both zones, Screen 0 and Screen 1, alternating each one per each column. That is , first column is in Screen 0, second in Screen 1, third in Screen 0, etc. In this mode all colours, including border, are with bright active (BRIGHT 1), and border colour is the same as the paper colour. Both screen parts use 6144 bytes each.

Radastan mode is a linear mode that uses just Screen 0, using each byte to define the colour of two pixels. This make up of this byte is defined (in binary) like this:

[AAAABBBB]

It means the leftmost pixel uses colour AAAA and rightmost pixel uses colour BBBB. The whole screen uses 128x96/2 = 6144 bytes again.

Activating the HiRes and the HiColour modes

Timex modes are both activated using port 0xFF (255). What you write to that port, is handled like this:

```
Bits 0-2:  Mode: 000=classic mode on screen 0, 001= classic mode on screen 1, 010 =
HiColour mode, 110=HiRes mode, other values are not valid.

Bits 3-5:  Only for HiRes mode, defines PAPER/BORDER and INK colours.
           000 – Black on white      100 – Green on magenta
           001 - Blue on yellow      101 - Cyan on Red
           010 - Red on cyan         110 – Yellow on blue
           011 – Magenta on green    111 – White on black

Bit 6:     If set to 1, the timer interrupt is disabled

Bit 7:     Selects bank to be used by the horizontal MMU0=DOCK, 1=EX-ROM.
```

Knowing that, the command required to activate HiColour is OUT 255, 2, and for HiRes is OUT 252, 6 – this will activate HiRes with black on white colour combination.

About the two upper bits, it's beyond the scope of this manual to fully describe them, so unless you know what you are doing, set them both to 0.

There is an alternative method of activating Timex modes using ULA, you can check how to do it, together with some other ULAplus features that we will explain later, here:
http://faqwiki.zxnet.co.uk/wiki/ULAplus

Activating Radastan mode

The Radastan mode is for the time being exclusive to ZX-Uno. It is a low resolution mode that doesn't have attribute clash. To activate radastan mode, using the RADASCTRL ZX-Uno register is needed (0x40, or 64). This is the sequence to activate it:

```
OUT 64571,64
OUT 64827,3
```

To go back to the previous mode, the second OUT value should be 0.

Note: the first version of the ZX Spectrum core used another method using the ULAPlus register. To avoid future incompatibilities that method has been removed.

There is a library for Z88DK ready to use the Radastan mode, the latest version so far can be found by following this link:
http://www.zxuno.com/forum/download/file.php?id=582

Changing video banks

As said above, it's possible to switch between Screen 0 and Screen 1, just selecting mode 000 or 001 on port 255, that is, use OUT 255,0 to choose Screen 0 at 0x4000, and OUT 255,1 to choose Screen 1 at 0x6000.
As the Timex modes and the radastan mode are independent of each other, you can also switch between two screens in radastan mode using the same OUT commands.

Going further, this paging system combined with bit 3 of port 0x7FFD, in case 128K paging is active, allows a total of 4 screens (Screen0, Screen1, Screen2 and Screen3) located at 4000h, 6000h, C000h and E000h.

Changing palette with ULAplus

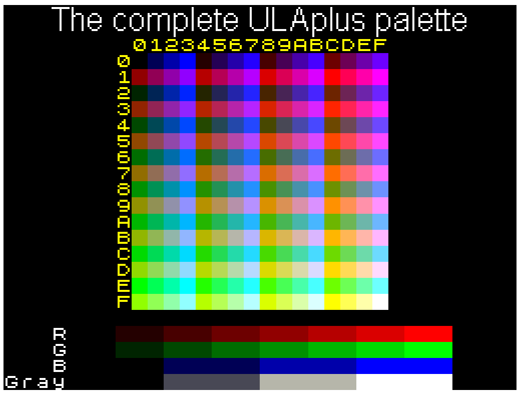
Using ULAplus we can use a 64 colour palette, but first we need to change to 64 colour mode like this:

```
OUT 48955, 64: OUT 65339, 1
```

Then we can modify each palette entry (0-63) assigning the colour X like this:

```
OUT 48955, n: OUT 65339, x
```

X is an RGB representation in 8 bits: GGGRRRBB. That is, 3 bits are used for green, 3 for red, and 2 for blue (the human eye is less sensitive to the colour blue):



The 64 bit palette is grouped in 4 groups of 16 colours, 8 for INK and 8 for PAPER (so INK and PAPER colours should not be the same ones). When you want to choose a colour for a specific 8x8 block the colour chosen depends on flash/bright/ink/paper like this:

	Spectrum colour:	0	1	2	3	4	5	6	7
Flash off Bright off	Ink	0	1	2	3	4	5	6	7
	Paper	8	9	10	11	12	13	14	15
Flash off Bright on	Ink	16	17	18	19	20	21	22	23
	Paper	24	25	26	27	28	29	30	31
Flash on Bright off	Ink	32	33	34	35	36	37	38	39
	Paper	40	41	42	43	44	45	46	47
Flash on Bright on	Ink	48	49	50	51	52	53	54	55
	Paper	56	57	58	59	60	61	62	63

Although in theory that would allow us to have 64 colours on screen at the same time, the truth is you only can do it if you paint bars, because it's impossible to use the INK colour from one group together with the PAPER colour from the other group.

We can redefine the palette for all modes mentioned above (standard, HiColour, HiRes and Radastan)

For radastan mode, the first 16 colours in the palette will be used to show the 16 available colours. At the time of writing this document, there is a plan for radastan mode to use colour #17 in the palette for the border, but that feature is not included in the current ZX Spectrum core version.

Raster interruption

The ZX Spectrum generates an interrupt with a frequency of 50Hz that is synchronized with the vertical retrace of the screen. This has often been used to make various graphic effects.

The ZX-Uno expands on this and allows activating what is called a “raster interrupt”. This new interrupt is generated at any selected screen line.

To use this interrupt, ZX-Uno registers RASTERLINE (0x0C) and RASTERCTRL (0x0D) should be set up first. The value in RASTERLINE combined with a 9th bit contained in the RACTERCTRL register will tell the ZX-Uno which line it should generate the interrupt on. This line is relative to the screen line where the paper area starts. The interrupt will be triggered when the hardware is about to paint the first pixel in the right border of the previous line to the one selected. This allows time for a machine routine to run before the hardware starts generating the screen output of the next line of the paper area.

For example, if the RASTERLINE value is 0 and RACTERCTRL register is 6, it will be triggered when the right border of the line just above the paper starts to be painted. If you want the interrupt to be triggered for line 257, set RASTERLINE to a value of 1 and set RACTERCTRL to a value of 7.

RACTERCTRL register is defined like this:

INT	0	0	0	0	DISVINT	ENARINT	LINE8

INT: this bit is read only. Its value is 1 during the 32 clock cycles after the raster interrupt has been triggered. This bit is available even if the processor has interrupts disabled. It is not available if ENARINT is 0.
DISVINT: set to 1 to disable the original maskable interrupts for the vertical retrace (standard ULA interrupt). After a reset this bit is reset to 0. So by default, standard ULA interrupts are enabled.
ENARINT: set to 1 to enable maskable interrupts produced by the raster interrupt. After a reset, this bit is reset to 0.
LINE8: Contains bit 8 (the MSB or the 9th bit) for RASTERLINE. This bit combined with RASTERLINE (register 0x0C) enables a value to be set between 0 and 511 inclusive. This defines the line where the interrupt occurs. Any value between 0 and 511 can be stored, although useful values are limited by the number of lines generated by the ULA (311 in 48K mode, 310 in 128K mode, 319 for Pentagon clone). If a value over the limit is set, the raster interrupt will never be triggered.

As you can see, to enable raster interrupt you need to explicitly activate it using ENARINT, and also when you do that, it’s normally common practice to disable the original interrupt with DISVINT. Unless we have a machine code routine that requires both of them to generate a special effect.

Raster interrupt may be used to make very advanced colour palette combinations, because although there are only 64 palette colours in the ULAPlus, nothing prevents us from changing them using the raster interrupt, allowing actually to use up to 64 colours per line.

It's even possible to switch video mode while the ULA is painting the screen, so it's actually possible to have half the screen in one mode and half the screen in another mode. Imagine for example a text adventure using HiColour for the upper part of the screen for a picture, and HiRes for the lower part for the text.

Turbo: boosting CPU speed

The ZX-Uno is able to run the CPU at 7 or 14MHz as well as the standard 3.5MHz. To activate those frequencies, modifying the Zx-Uno SCANDBLCTRL (0x0B) register is required.

If you just want to test it, just change speed from the BIOS or use ZXUC after the ZX-Uno has booted. There is a chance that with some BIOS versions you find a 28MHz mode, but in that case take into account that this is an experimental mode that may not work well in a real ZX-Uno.

In order to boost your software speed, have a look at how the 0X0B register is configured (leftmost bit is bit 7)

TURBO	0	FREQ	ENSCAN	VGA

Bits 7-6 set up turbo mode, where 00 is for 3.5 MHz, 01 for 7 MHz, and 1x for 14 MHz.

It may look like setting the register to 0 sets normal mode, 64 for 7MHz and 128 for 14MHz

```
OUT 64571,11      (0X0B)
OUT 64827,128
```

But actually, you have to take into account that bits 0-1 control the way the output to screen is handled, controlling the VGA or PAL connection. So it is best to leave those settings, and also the FREQ setting as they are. Otherwise you may mess things up so that it is not possible to see anything on the screen.

The best way to set turbo mode is first read the current value from the register:

```
OUT 64571,11   (0X0B)
LET VAL = IN 64827,128
```

Then clear bits 6-7 (you can do that with AND 63 in assembler, but as Sinclair basic doesn't have a bitwise AND operator you can do it like this:

```
IF (VAL>=128) THEN LET VAL=VAL-128
IF (VAL>=64) THEN LET VAL=VAL-64
```

And finally add 0 for normal mode, 64 for 7MHz or 128 for 14MHz, and modify the register again:

```
LET VAL=VAL+64
OUT 54571,11
OUT 64827,VAL
```

Mouse

The ZX-Uno implements a Kempston mouse interface via the PS/2 connector, where we can connect a PS/2 mouse using a splitter (so we can connect keyboard too).

Mouse data is obtained by Reading several ports:

Port	Hex	Value
64479	FBDF	Horizontal position (X-Axis)
65503	FFDF	Vertical position (Y-Axis)
64223	FADF	Button status (bitwise)

That way, PRINT IN 64479 will return the X-Axis value.

Value of port 64223 can be decoded like this

Bit	Content
0	0 if right button is pressed
1	0 if left button is pressed
2	0 if central button is pressed
3	0 if fourth button is pressed
4 to 7	Returns mouse wheel position, defaults to 1111

Note: keep in mind there is no driver supporting the mouse, so don't expect to find a mouse pointer on screen when Zx-Uno boots. Programs that use the mouse will paint the pointer on screen. If you want to support a mouse in your software you will have to paint the pointer yourself.

As well as reading the mouse as a Kepston mouse, we can also read data received through the PS/2 port directly using MOUSEDATA (0x09) and MOUSESTATUS (0x10) registers. Describing that is outside the scope of this manual, but you should know it's possible and in case you need it you surely will find details of how PS/2 mice work on the internet.

Sound AY Chip

The AY chip is available in the ZX Spectrum 128K models, but in the ZX-Uno it's available even if you boot a 48K ROM. Of course, if you boot with a 128 K ROM you can use PLAY command from BASIC, but for more advanced tasks you'll need to know this:

The AY chip has several programmable registers:

Register	Function	Range
0	Channel A fine pitch	8-bit (0-255)
1	Channel A coarse pitch	4-bit (0-15)
2	Channel B fine pitch	8-bit (0-255)
3	Channel B coarse pitch	4-bit (0-15)
4	Channel C fine pitch	8-bit (0-255)
5	Channel C coarse pitch	4-bit (0-15)
6	Noise pitch	5-bit (0-31)
7	Mixer	8-bit (see below)
8	Channel A volume	4-bit (0-15, see below)
9	Channel B volume	4-bit (0-15, see below)
10	Channel C volume	4-bit (0-15, see below)
11	Envelope fine duration	8-bit (0-255)
12	Envelope coarse duration	8-bit (0-255)
13	Envelope waveform	4-bit (0-15)
14	I/O port A	8-bit (0-255)
15	I/O port B	8-bit (0-255)

To modify a register you should write the register number on port 65533, and then write the value to set on port 49149. To read a register value (not very common, but possible) you have to write the register number to port 65533 and then read again port 65533 (not 49149).

For instance this code sets 255 on register 7:

```
OUT 65533,7
OUT 49149,255
```

The mixer register has several fields, that allow activating or deactivating several channels:

7	6	5	4	3	2	1	0
I/O	I/O	Noise C	Noise B	Noise A	Tone C	Tone B	Tone A

The I/O ports are not used in the ZX-Uno right now, but there is a chance that they will be used in the future.

Pitch values

To determine the frequency of the waveform on each channel, that will define the tone played, the pitch values are used. Each tone has a specific frequency assigned, and any tone out of those is considered off-key. This is a list of common frequencies for middle octave:

Tone	Frequency (Hz)	Tone	Frequency (Hz)
A	220	D#	311.1
A #	233.3	E	329.63
B	246.94	F	349.23
C (middle)	261.63	F#	370
C#	277.2	G	392
D	293.66	G#	415.3

The AY chip won't allow us to directly specify the frequency though, instead of that we will define the pitch, that is a value from 1 to 4095, and the frequency out of a channel is the one resulting from dividing 110,83KHz with the pitch value. So we can obtain a value from 27Hz (pitch 4095) to 110.83Khz (pitch 1).

Pitch value is selected using two registers, the one named "coarse" and the one named "fine". It's a 12 bit value and the coarse register contains the 4 most significant bits (value 0-15) and the fine pitch the lower 8 bits.

A simple example in BASIC:

```
OUT 65533,0 :REM Fine Pitch channel A=43
OUT 49149,43
OUT 65533,1 : REM Coarse Pitch channel A=1
OUT 49149,1
OUT 65533,8 : REM Volume channel A   = 12
OUT 49149,12
OUT 65533,7 : REM MIXER, active channel A
OUT 49149,1
```

So with a coarse pitch=1 and fine pitch=43, the pitch value is 299. 110,83Khz divided by 299 equals 370Hz, that is a F#, that will be played with a volume of 12 (out of 15) through channel A as mixer says channel A is active.

Take into account that due to technical limits, it's more difficult to obtain accurate frequency values in the lower frequencies, so tones may be slightly off-key on those tones.

Volume

Registers 8, 9 and 10 contain a volume value in the lower 4 bits, so volume can be any value from 0 to 15. Despite of that, if bit 5 is active, then volume is ignored and an envelope wave defined in register 13 is applied.

Envelope wave

AY chip supports envelope waves defined by register 13. Please note this affects wave amplitude (volume) but not frequency (tone).

Wave applied is defined by four values in register 13:

Bit	Command
3	Continue
2	Attack
1	Alternate
0	Hold

Depending on the active bits one or another wave will be applied.

The frequency of the enveloping wave depends on register 11 and 12 values. Both registers define a 16 bit value, the coarse one is the most significant byte. That value will divide a base frequency of 6,91KHz, so if the value is 1, the wave will have a 6,91KHz frequency, and if the value is 65535, the frequency would be 0,11Hz.

You can find extended information about AY chips in the following document: http://dev-docs.atariforge.org/files/GI_AY-3-8910_Feb-1979.pdf

Turbo Sound

The ZX-Uno supports Turbo Sound, that basically is a second AY chip. By default the first AY chip is accessed via the standard registers, but if we want to switch to the second AY chip we just need to use OUT 65533,254. Then all the AY registers modified will be on the second AY chip. To get back to using the first AY chip use OUT 65533, 255.

Accessing the SD card with ESXDOS

Information about ESXDOS is not public, but some calls are known about. The information below is valid for ESXDOS 0.8.6, and may not work in other versions, or even in all cases, though all content in this manual should be used at your own risk, take in mind this is the trickiest section:

The first thing you need to know, is that all ESXDOS calls are made via RST 8, and just after the RST 8 call, the code of the function called should be placed (with a DB). Some functions will require also having some values specified in some CPU registers:

Detect if unit is ready

```
M_GETSETDRV   equu   $89
XOR a
RST $08
DB M_GETSETDRV
```

Carry flag will be set if error.

Open file

```
F_OPEN   equ   $9a
XOR a
LD B, FA_READ   ; b = open mode
LD IX, FileName   ; ix = Pointer to file name (ASCIIZ)
RST $08
DB F_OPEN      ; open read mode
```

Returns a file handler in 'A' register.

Modes to open a file:

```
FA_READ   equu   $01
FA_APPEND equ $06
FA_OVERWRITE equ $0C
```

Read from file

```
F_READ   equ   $9d
LD IX, 16384   ; ix=address where to store what is read
LD BC, 6912   ; bc=bytes to read
LD A, handler; A = the file handler
RST $08
DB F_READ      ; read file
```

Carry flag is set if read fail.

Write to file

```
F_WRITE   equ   $9e
LD IX, 16384   ; ix=memory address to write from
LD BC, 6912   ; bc=bytes to write
LD A, handler; A = file handler
RST $08
DB F_WRITE     ; write file
```

Carry flag is set in case of write fail.

Close file

```
F_CLOSE   equ   $9b
LD A, handler ; A =file handler
RST $08
DB F_CLOSE
```

Carry flag active if error when closing

Notes

File paths use the slash character (‘/’) as directory separator (UNIX style).

ZX-Uno registers

The ZX-Uno provides several registers that allow you to configure ZX-Uno properly, and activate or deactivate some peripherals. We have used some of them in the sections above. There are two ports $FC3B (64571) and $FD3B(64827) assigned by the ZX-Uno team. From them you can access up to 256 registers (though not all of them are used… yet).

- Port 0xFC3B (64571) saves the address (0x00 – 0xFF) of the register to read/write.
- Port 0xFD3B (64827) is the data port and reads or write the register selected by the other port.

For example, to write a value X to register 0x40, use OUT 64571,64: OUT 64287,X, while to read a value from register 0x00 we do OUT 64571,0: LET X = IN 64287

Some registers can only be read or written, and some others are read/write registers.

In this manual, only details of registers from 0x00 to 0x0F, register 0x40 and registers 0xFC to 0xFF are given. You can find detailed information in this listing:

http://www.zxuno.com/wiki/index.php/ZX_Spectrum#Nuevos_registros_E.2FS_para_control_de_ZX-Uno

Register summary:

#	Name	R/W	Description
$00	**MASTERCONF**	R/W	Allows enabling or disabling some features (DivMMC, NMI de DIvMMC, boot mode , ULA timing (to accommodate to 48k, 128k or Pentagon timings) and contended memory.
$01	**MASTERMAPPER**	R/W	Memory map, that defines which SRAM bank will be mapped at $C000-$FFFF. Although ZX-Uno doesn't have that much memory, up to 4Mb of RAM can be mapped this way.
$02	**FLASHSPI**	R/W	SPI flash Access port.
$03	**FLASHCS**	R/W	Value written on bit 0 determines the status of CS line on the SPI flash (0 flash selected, 1 flash not selected)
$04	**SCANCODE**	R/W	Returns last scancode received form the PS/2 keyboard
$05	**KEYSTAT**	R	Several bits that detail if there is a new key pressed, released, extended key or normal, etc.
$06	**JOYCONF**	R/W	Controls the joystick emulated by the ZX-Uno, may be Sinclair 1, Sinclair 2, Kempston or Cursor (AGF/Protek).
$07	**KEYMAP**	R/W	Allows Reading keyboard map and modify it
$08	**NMIEVENT**	R	Returns last NMI cause
$09	**MOUSEDATA**	R/W	Mouse data registered, allow Reading or sending data
$0A	**MOUSESTATUS**	R/W	Mouse status record
$0B	**SCANDBLCTRL**	R/W	Scan double control and turbo modes
$0C	**RASTERLINE**	R/W	Saves the least significate bits of the raster line that will trigger a raster interrupt
$0D	**RASTERCTRL**	R/W	Controls the raster interrupt (and vertical retrace interrupt)
$0E	**DEVCONTROL**	R/W	Allows enabling and disabling some features (interface hardware SPI, MMU horizontal Timex, etc.)
$0F	**DEVCTRL2**	R/W	Allows enabling and disabling some features (Radastan, modes Timex and ULAPlus)
$40	**RADASCTRL**	R/W	Controls the Radastan video mode
$FC	**COREADDR**	E	Stores the position of second core in the SPI
$FD	**COREBOOT**	E	Controls boot
$FE	**SCRATCH / COLDBOOT**	R/W	This just stores a 8 bits value, ZX-Uno makes this value to 0 after power on, it won't be zero again after a reset or master reset,
$FF	**COREID**	R/W	Successive readings will return an ASCII string containing the CORE version.

Part IV – Advanced options

Installing Spectrum ROMs via cassette deck

In order to add a new ROM so it's available from the ROM selection menu at boot time, you should first convert that ROM into a TAP file. That should be played later through the EAR input connector (using tapir or any other TAP file playing utility).

In case you don't have the ROM file in TAP format, you can use **GenRom.exe** located at the firmware/roms folder at the ZX-Uno repository:

http://guest:zxuno@www.atc.us.es/svn/zxuno/firmware/roms/

You can check which program parameters are set at the command line. Although those parameters are quite technical, they match the contents of the ZX-Uno registers described above, although details are better described at the ZX-Uno wiki (in Spanish):

http://www.zxuno.com/wiki/index.php/ZX_Spectrum

Once you have the tap file ready, enter the BIOS at boot time, then go to ROMS menu.

- If you want to add a new ROM, we press N and we will be asked which slot will we be using. We can choose any slot from 0 to 63, but be aware that each slot takes 16K. So if the ROM is larger (i.e. +2/toastrack ROM is a 32K ROM, and +2A/B/+3 is a 64K ROM) it will take up more than one slot. Be careful, because if you choose a slot number without enough empty slots after it and ROM is larger than 16K, the ROMS after that slot may be overwritten.
- If you want to replace a ROM, just press intro over that ROM, and choose delete. Once deleted use the option to add a new ROM.

You will then be requested to play the ROM tap file through the EAR input.

Installing Spectrum ROMs via SD card

Providing access to SD card from BIOS was a complicated process considering the ZX-Uno architecture, so to load ROMs from SD card won't be as easy as one would expect. Anyway this is the process required.

At same folder where GenROM.exe is located, there is also another tool named generacms.bat, and a lot of ROMS. Download all the contents of that folder (including sub-folders).

http://guest:zxuno@www.atc.us.es/svn/zxuno/firmware/roms/

If we have a look at generacms.bat, we can see this:

```
call :CreateRom 0  "ZX Spectrum 48K Cargando Leches" leches         dnlh
call :CreateRom 1  "ZX +2A 4.1"                     plus3en41       t
call :CreateRom 5  "SE Basic IV 4.0 Anya"          se              dh
call :CreateRom 7  "ZX Spectrum 48K"               48              dnlh17
```

Ok, each line has the slot number on third column, followed by ROM description between double quotes, and fifth column is the filename that contains that ROM without the extension.

Finally, last column are the options available in GenROM.exe.

Well, what we are going to do is to replace all the ROMs all together. To do that what we should do is replace whatever is in that file, or add new ROMs, and then run generacms.bat. That will create a file named ROMS.ZX1 in the sd_binaries sub-folder. We should copy that file into the SD card root folder.

Apart from that file, we should add the ROMSUP utility in the BIN folder of the SD card. It's located here:

http://guest:zxuno@www.atc.us.es/svn/zxuno/software/spiflashutils/

Once we have that ready, we are going to update all ROMs by booting Zx-Uno, pressing (Break, Caps+Space, Esc) on boot screen, choosing "rooted" ROM and after we get to basic type:

.romsup

The ROMS update will be executed. The "Rooted" ROM allows writing the ROMS, other ROMs have that option disabled.

Installing or updating cores

To update or install a new core for the **ZX-Uno** there are two options:

First, and probably the more complicated, is generating the .bit file and write it in the FPGA using the Xilinx cable and iMPACT software. Although this process is faster, we need to install software, and buy a relatively expensive cable. Also, we will lose that update after turning ZX-Uno off unless we generate the MCS file for the SPI.

The easiest way of updating cores is generating a TAP file and loading via BIOS. To do so, we should enter the BIOS, go to "UPGRADE" menu, choose an empty core slot or an existing core, and press Intro. At this moment we should play the TAP file via EAR connector. TAP files are available at this Google Drive account.

If everything works as expected, you will see the usual loading border and status will be updated. It will take some minutes to upgrade cores.

Generating the firmware

Warning: this is for very advanced users, as special cable is needed to save the SPI memory.

Download the Project Repository http://www.atc.us.es/svn/zxuno/ (user: guest / password: zxuno) using Subversion. We could use Tortoise SVN from Windows.

Download and install ISE Webpack 14.7 from the Xilinx official website.

In the firmware/roms folder, edit **promgen.bat** to point to our Xilinx software path.

Finally, edit the file **generamcs_multiboot_Q??.bat so** that it matches our board (Q80 for first prototypes, Q32 since ZX-Uno v3.0)**,** also checking th cores to include and their description/names. Running that file, a **prom_multiboot_Qnn.mcs** file should be created, where again nn depends on the SPI flash of our ZX-Uno.

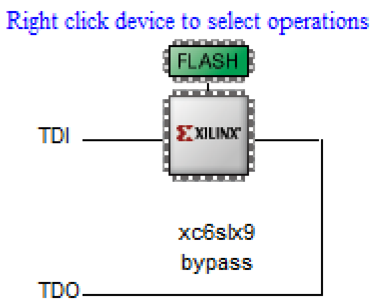
Saving and restoring the SPI

First connect the **Xilinx Platform Cable USB** to our PC and **ZX-Uno**, power on **ZX-Uno** and open **iMPACT** software. Then create a new Project, and let the software detect the FPGA.

If it fails when selecting files (*Windows 10 64 bits*), do the following:

- At INSTALLATION_PATH\Xilinx\14.7\ISE_DS\ISE\lib\nt64 rename **libPortability.dll** to libPortability.dll.old and **libPortabilityNOSH.dll** to libPortability.dll
- Copy the new **libPortability.dll** to INSTALLATION_PATH\Xilinx\14.7\ISE_DS\common\lib\nt64 (make a backup of the file to be replaced first if you prefer).

Then right click on the chip associated to the FPGA, and choose "*Add SPI/BPI flash*". Then browse to the MCS file we generated (**prom_multiboot_Qnn.mcs**), choose SPI type (**W25Q32BV** for ZX-Uno above v3 or W25Q80BV for first v1 and v2 prototypes), and set bus size to 1.



If we want to write the file just choose "Program", uncheck "Verify" in the popup window, and press OK

To make a backup, just choose "Readback…", provide a name for file and press OK.

Unbricking the ZX-Uno

There is a chance that while updating ROMS or CORES, because of a mistake (writing wrong file), or because of a mistake made by someone else (writing wrong file provided by someone else), or because some external failure (power failure while updating) we end up with a *bricked* ZX-Uno that won't work. If we have a Xilinx cable described above, we can use it to bring back our ZX-Uno to life, but there are other cheaper options:

Altera cloned cable from Windows

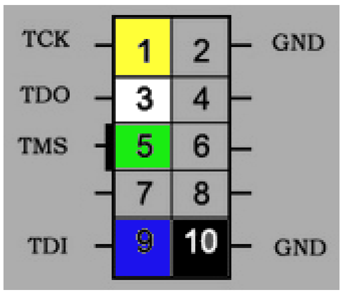
For just 3.50€ you can buy this cable at Aliexpress, it may take between one and two months to arrive (to Europe at least).

http://es.aliexpress.com/item//1487146376.html

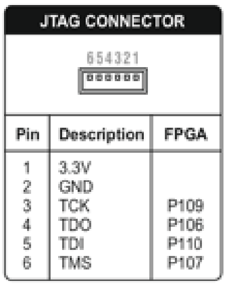
We will also need a microJST connector for the JTAG connector in our ZX-Uno, a connector for the 10 pin cable, and some wires to solder the adapter cable.

10 pin conector Altera cable

This is the way the wires should be connected to the 10 pin cable on the Altera side:



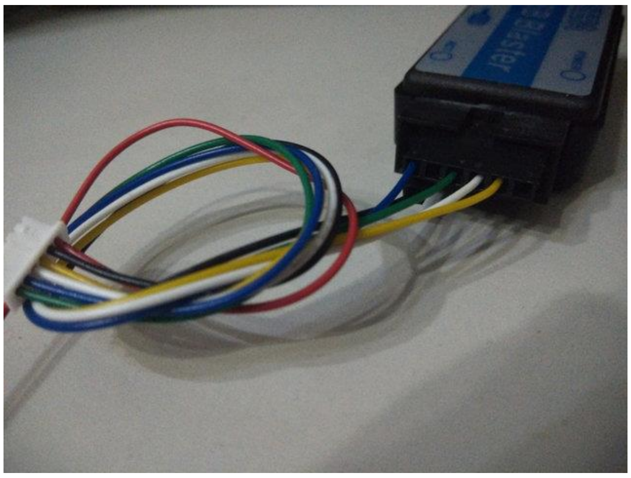
And this is the JTAG connection pinout:


JTAG CONNECTOR

654321

Pin	Description	FPGA
1	3.3V	
2	GND	
3	TCK	P109
4	TDO	P106
5	TDI	P110
6	TMS	P107

In the end we should get something like this:



Note: this picture has a 12 pin connector, that's why the rightmost pins are not used. As you can see the wires are short. It's not recommended to build the cable with long wires as it has been found they won't work in some cases.

Once we have built the cable, we need to get the urjtag software and Windows drivers, together with the file needed to restore boot. Easiest way to get that is downloading from zxuno.com forum:

http://www.zxuno.com/forum/download/file.php?id=505

Now connect the ZX-Uno to your PC using the cable and USB Blaster, and power on the ZX-Uno.

After decompressing the downloaded file, go to Device Manager in Windows and find the USB Blaster (as unknown device). We choose the option to add drivers manually and browse to the folder where they are. If it works, we should then see "Alter USB-Blaster" in the Device Manager.

Now open a command line, and go to the folder where jtag is and type:

```
jtag
```

Once we are into jtag software, we type:

```
cable usbblaster
detect
```

That "detect" command should show a message with the FPGA model.

Then we go for the following command:

```
pld load recovery.bit
```

In the screen the BIOS menu should appear, then insert a SD card with the FLASH.ZX1 described before in the root folder and press Intro.

The current FLASH.ZX1 file can be found in the repository:

http://zxuno.speccy.org/ficheros/4.1/flash.zip

Using a Raspberry Pi

If you own a Raspberry Pi with Raspbian installed, you can also restore your ZX-Uno without the Altera cable. Follow these steps to do it:

Download the file described in the previous section and get the recovery.bit file. Save it in the SD card (for instance in the /boot folder that can be accessed from Windows).

From the Raspberry Pi, and assuming it's a model A+, B+ or above, and we are using Raspbian, do this:

```
sudo apt-get install autoconf
sudo apt-get install autopoint
sudo apt-get install libtool
sudo apt-get install libreadline-dev
sudo rm -rf /var/lib/apt/lists/*
sudo apt-get update
sudo apt-get install python-dev
sudo apt-get install git
```

Now download urjtag from its Git repository:

```
git clone git://git.code.sf.net/p/urjtag/git urjtag-git
cd urjtag-git/urjtag
```

Now edit the file src/cmd/cmd_bfin.c with your favourite Linux editor:

```
nano src/cmd/cmd_bfin.c
```

and add the following as the first line:

```
#define _SYS_UCONTEXT_H
```

Finally compile urjtag using the following commands:

```
./autogen.sh
make
sudo make install
```

Once compiled, we should run ldconfig and jtag:

```
sudo ldconfig
sudo jtag
```

Then turn on the ZX-Uno, insert the card with the FLASH.ZX1 file in root folder, and flash the ZX-Uno. Connect the ZX-Uno to the Raspberry Pi using the diagram below.

Once everything is connected, run the following commands:

```
cable gpio tdi=13 tdo=19 tck=26 tms=6
detect
```

If it worked, it will “detect” and will return to the FPGA model. Now run the following command (assuming recovery.bit file is at /boot folder):

```
pld load /boot/recovery.bit
```

Then wait a while until we see the upgrade menu, and press intro in the ZX-Uno.

This is the pinout for the cable that connects the ZX-Uno to the Raspberry Pi GPIO connector:

ZX-Uno		Raspberry Pi	
JTAG	Pin	GPIO	Pin
TMS	6	GPIO6	31
TDI	5	GPIO13	33
TDO	4	GPIO19	35
TCK	3	GPIO26	37
GND	2	GND	39
3.3V	1	N/C	N/C

Please notice the 3.3v from ZX-Uno is not connected.

Part V - Cores

ZX Spectrum

The ZX Spectrum core implements the following:

- Implements a ZX Spectrum 48K, 128K, Pentagon and Chloe 280SE
- ULA with ULAplus, Timex and Radastan modes
- Supports ZXMMC for +3e
- Supports DIVMMC with ESXDOS 0.8.6 beta4 and above, this means:
  - You can load software from SD card in SNA, Z80 or TAP formats
  - Advanced commands for accessing SD card (ls, cd, mkdir, mv, etc.)
  - Virtual tape simulation with ftap files and extended commands (.tapein, .tapeout)
  - Virtual disk support with .TRD files format and TR-DOS.
  - Possibility to work as a +3e system
- AY Chip support for music, also in 48K mode
- Turbo Sound support (second AY chip)
- Joystick support, and second joystick emulated with keyboard support (Kempston, Sinclair 1, Sinclair 2, or Cursor/Fuller)
- Turbo mode support, with CPU speed of 7MHz or 14MHz
- Keymap support for PS/2 keyboards, configurable from the ZX Spectrum core.
- PS/2 mouse support, emulating Kempston Mouse protocol.
- Option for composite mode output, RGB 15KHz or VGA.
- Vertical refresh frequency editable to improve compatibility with VGA display units.
- 512k RAM
- Raster interrupt, allowing synchronization of actions during screen paint, for instance changing palette what may lead to a 256 colour picture on screen.
- Option to disable memory contention for Pentagon 128 compatibility.
- Option to select keyboard model (issue 2 or issue 3)
- Option to choose ULA timing (48K, 128K or Pentagon)
- Option to enable/disable the memory bank control, to improve compatibility with some models.
- Option to disable ULAplus, Radastan mode and Timex modes, etc. for better compatibility with some games.

Other cores

The main target of the ZX-Uno Project is to implement a ZX Spectrum, so this core is the one with best information and best implementation. Despite that, other cores have been adapted to work with ZX-Uno.

Currently these cores are available:

Computers

- **SAM Coupé**. Supports keyboard, loading via EAR connection, and has composite video, RGB and VGA output.
- **Jupiter Ace**. Supports keyboard, loading via EAR connection, and has PAL and VGA output.
- **Apple][**. For the time being it only supports VGA output, and disk should be loaded formatting the SD card in a special format. Allows using two button joysticks.
- **Acorn Atom**. Only VGA output, but joystick support is provided and allows loading software via SD card.
- **BBC Micro**. No joystick or EAR load support, but supports all video modes and loading software via SD card with MMB images.
- **Acorn Electron**. Similar to BBC micro, but supports EAR loading.
- **Oric Atmos**. No support for loading software via EAR or SD card, but you can type in programs manually. VGA and PAL output.
- **VIC-20**. No support for loading software, only VGA output.

Consoles

- **Master System**. It implements a ROM selector (they can be located anywhere in the SD card), support SDHC cards, joystick, and PAL/VGA output.
- **NES**. Only VGA output supported, the implemented console is NTSC. Can load ROMs from SD card, it supports joystick and some common mappers.
- **Atari 2600**. Allows loading ROMs from SD card, but it has some graphic malfunctions and only VGA output is supported.

Special keys for other cores

Each core has some special key combinations that are used for specific functions. On some cores the Caps Lock key rotates between PAL and VGA mode, but not on all of them.

BBC Micro

PgDw, PgUp	Changes video mode (VGA/Composite)
Shift+F12	File browser (hold shift key)
F12	Reset

Acorn Atom

Shift+F10	File browser
F10	Reset
F1-F4	Turbo modes (F1 = 1MHz, F2 = 2MHz, F3 = 4MHz, F4 = 8MHz)

Acorn Electron

PgUp, PgDown	Video mode VGA/Composite
F10	Reset

Master System

F12	Browser ROM
Pause	Pause

Atari 2600

Esc	Show menu

Apple II

Shift + Fx	Run disk (x= number of the disk to run)

Preparing SD card for other cores

Each core may have different requirements about SD card contents or format. In some cases two different cores may share the same card, but others will need a specific card.

Acorn Atom

Format the card as FAT and include these files in the root folder:

http://www.stardot.org.uk/forums/viewtopic.php?f=44&t=6544

BBC Micro / Acorn Electron

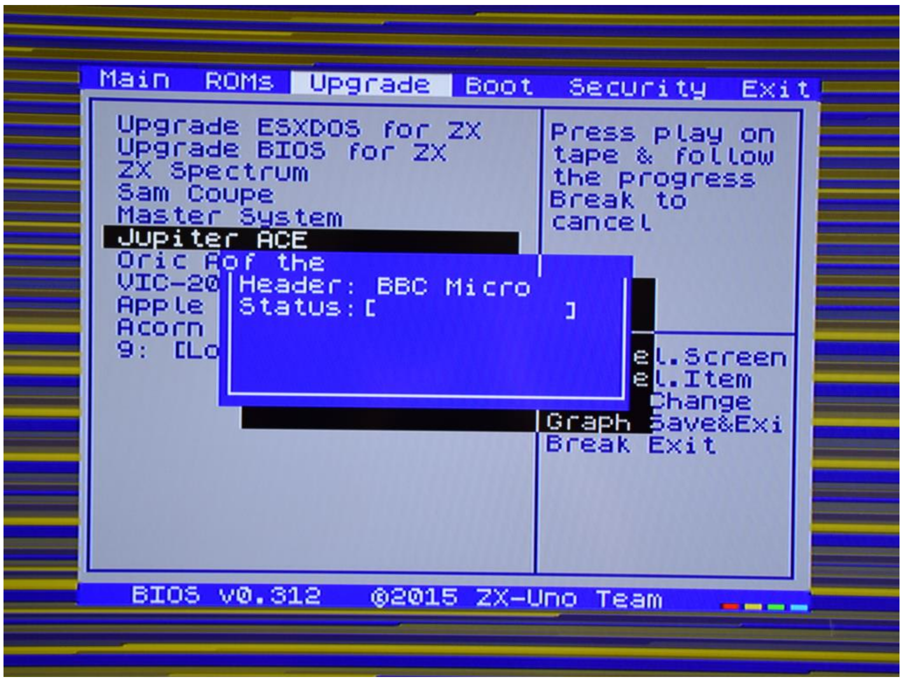
Card should be formated without using fast format, and prepare a BEEB.MMB file in the root folder with MMBImager.exe including it in the card first (don't copy any other file beforehand).

https://swhs.home.xs4all.nl/bbc/mmbeeb/windows.html

Apple II

We need the following:

- A disk image in .nib format, you can use dsk2nib.exe avaliable at ROMS folder to convert it. There are some images ready in that folder (DOS 3.3 , Apple33.nib and Spy vs. Spy game, SpyVsSpy.nib)
- A dedicated SD card. All content will be erased.
- An utility to copy RAW disks as Linux **dd** de GNU/Linux or HDD Raw Copy Tool for Windows, that will copy the nib file into the SD card.


Main ROMs Upgrade Boot Security Exit

Upgrade ESXDOS for ZX
Upgrade BIOS for ZX
ZX Spectrum
Sam Coupe
Master System
Jupiter ACE
Oric Aof the
VIC-20
Apple
Acorn
9: [Lo

Header: BBC Micro
Status: [ ]

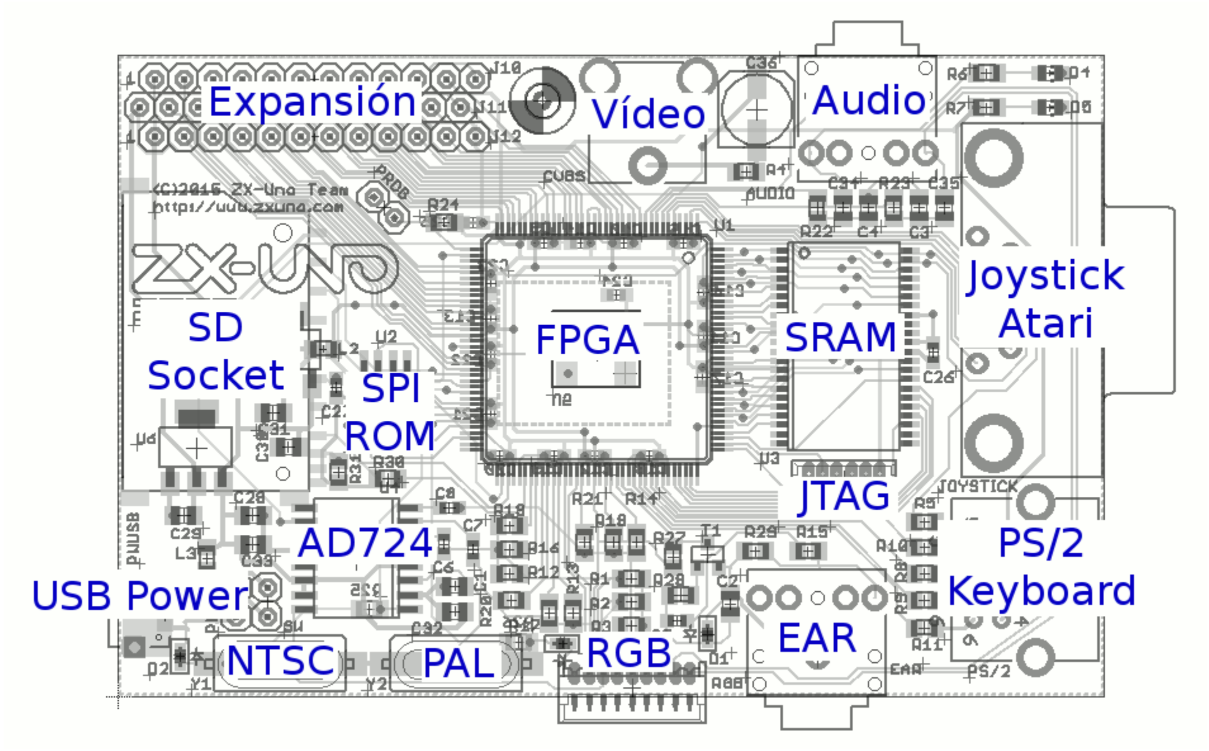
Press play on tape & follow the progress Break to cancel

el.Screen
el.Item
Change
Graph Save&Exi
Break Exit

BIOS v0.312 ©2015 ZX-Uno Team

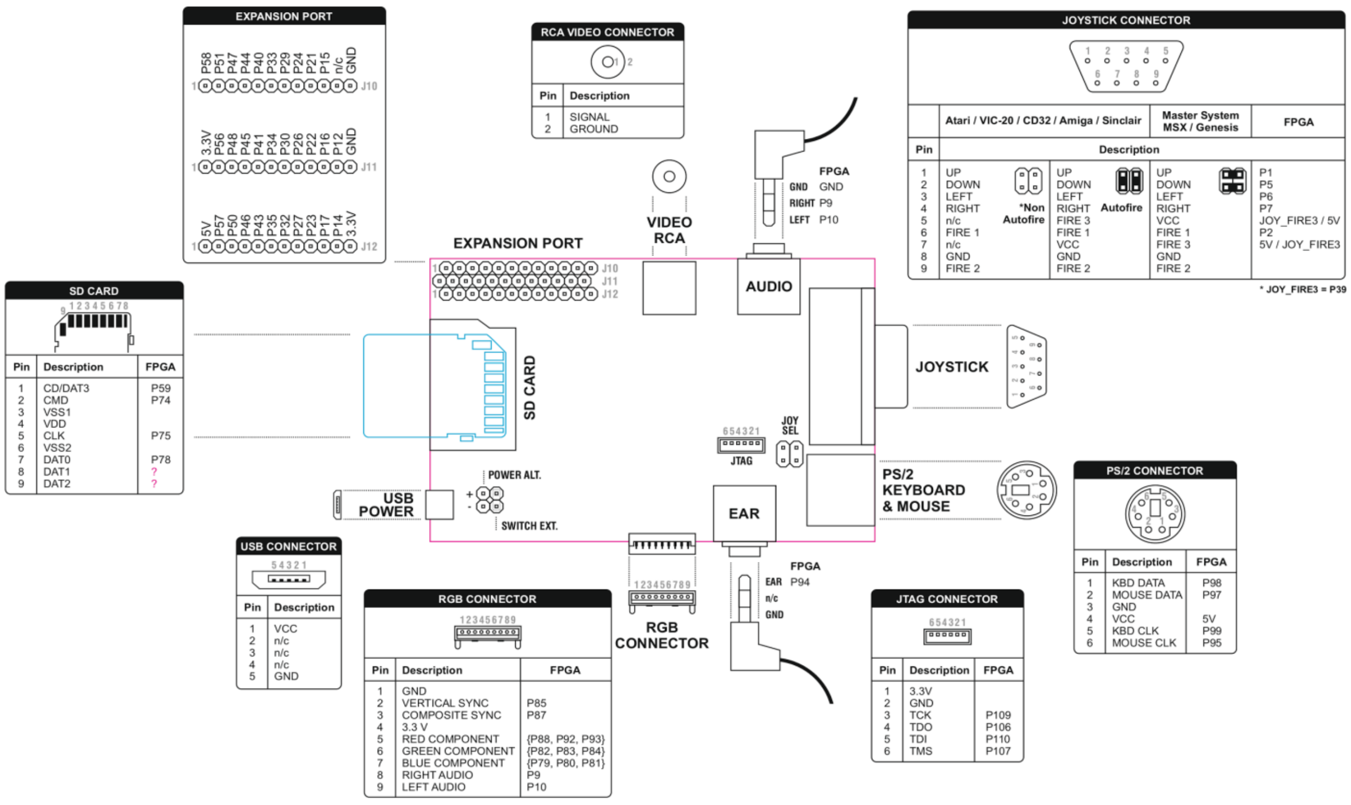
Part VI - Appendix

ZX-Uno board layout



Pinouts

In the following picture, you can see the pin outs for all the connectors used on the ZX-Uno. To use the expansion port for peripherals, you have to consider its using 3.3V, while Spectrum peripherals use 5V, so adapting an existing peripheral will require a level converter.



EXPANSION PORT

	1	2	3	4	5	6	7	8	9	10	11	12
J10	P58	P51	P47	P44	P40	P33	P29	P24	P21	P15	n/c	GND
J11	3.3V	P56	P48	P45	P41	P34	P30	P26	P22	P16	P12	GND
J12	5V	P57	P50	P46	P43	P35	P32	P27	P23	P17	P14	3.3V

RCA VIDEO CONNECTOR

Pin	Description
1	SIGNAL
2	GROUND

AUDIO

	FPGA
GND	GND
RIGHT	P9
LEFT	P10

JOYSTICK CONNECTOR

	Atari / VIC-20 / CD32 / Amiga / Sinclair		Master System MSX / Genesis	FPGA
Pin	Description			
	*Non Autofire	Autofire		
1	UP	UP	UP	P1
2	DOWN	DOWN	DOWN	P5
3	LEFT	LEFT	LEFT	P6
4	RIGHT	RIGHT	RIGHT	P7
5	n/c	FIRE 3	VCC	JOY_FIRE3 / 5V
6	FIRE 1	FIRE 1	FIRE 1	P2
7	n/c	VCC	FIRE 3	5V / JOY_FIRE3
8	GND	GND	GND	
9	FIRE 2	FIRE 2	FIRE 2	

* JOY_FIRE3 = P39

SD CARD

Pin	Description	FPGA
1	CD/DAT3	P59
2	CMD	P74
3	VSS1	
4	VDD	
5	CLK	P75
6	VSS2	
7	DAT0	P78
8	DAT1	?
9	DAT2	?

USB CONNECTOR

Pin	Description
1	VCC
2	n/c
3	n/c
4	n/c
5	GND

EAR

	FPGA
EAR	P94
n/c	
GND	

RGB CONNECTOR

Pin	Description	FPGA
1	GND	
2	VERTICAL SYNC	P85
3	COMPOSITE SYNC	P87
4	3.3 V	
5	RED COMPONENT	{P88, P92, P93}
6	GREEN COMPONENT	{P82, P83, P84}
7	BLUE COMPONENT	{P79, P80, P81}
8	RIGHT AUDIO	P9
9	LEFT AUDIO	P10

JTAG CONNECTOR

Pin	Description	FPGA
1	3.3V	
2	GND	
3	TCK	P109
4	TDO	P106
5	TDI	P110
6	TMS	P107

PS/2 CONNECTOR

Pin	Description	FPGA
1	KBD DATA	P98
2	MOUSE DATA	P97
3	GND	
4	VCC	5V
5	KBD CLK	P99
6	MOUSE CLK	P95

Useful links

The official repository:

http://guest:zxuno@www.atc.us.es/svn/zxuno

The ZX-Uno wiki:

http://www.zxuno.com/wiki
http://www.zxuno.com/wiki/index.php/ZX_Spectrum

The Google Drive account:

https://drive.google.com/folderview?id=0B_p-bMWLxui7WEFuWUFvOEVOMEE