ZXSpectr Version 3.4

ZX Spectrum Emulator by César Hernández Bañó

01-12-2008

Index

1 History

Since I was a child, there was a computer at home; I've had many, and many of them are Sinclair. The first we had was a Sinclair ZX-81. It had a flat keyboard, it was tiny and it heated a lot. It worked only with 1K of RAM (that we expanded to 16k), and the CPU was the Z-80. It saved the data on tape, at the minimum speed of 300 bauds.

Later, we bought the ZX Spectrum 48k, with rubber keyboard. It was a very useful computer, although we changed 3 times the keyboard membrane and the heat controller.

In 1984, when the Spectrum had success, Sinclair presented a mega-computer, the QL, with the Motorola 68008 CPU, and we also bought it. People in Spain who bought a QL know that programs for QL were sold until Amstrad bought Sinclair.

After this, we bought 2 Inves Spectrum+, which were like a Sinclair Spectrum+ but made by Investronica; the trouble with this computer was that the sound wasn't heard in many games (you could fix it POKEing in the ROM!). After a long time I managed to get a Spectrum +2A, changing a Inves.

And in the PC times, I was still programming with my +2A, and one day I get my first emulator, the SPECTRUM of Pedro Gimeno, and later, the Z80 of Gerton Lunter. Since then, I always wanted to make my own emulator and load my old games without buying any emulator or making any electric circuit.

In 1996 I made the first version, in Assembler, and using the ROM of Pedro Gimeno's Emulator and the SP files. And two years later, in 1998, we bought a Pentium 133 with Sound Blaster card and I could make a program to load games directly from tape. Then I loaded the ROMs of my +2A and I added the 128k and the AY sound to the emulator.

Since then I've been making new versions of the emulator and sending it to my friends, and version 1.5 was the first one that I sent to Internet.
It doesn't work at 100% with all games, but I haven't found the bug yet. It would be very grateful that someone who find the bug tells me. I think the bug is related with the P/V flag.

Nevertheless, the emulator works fine, it has many options that hasn't another emulators, you can load games from a real tape and it's totally free.

Maybe this is the last version of my emulator for MS-DOS; from version 3.3 I've been compiling and testing the emulator in a MS-DOS emulator for Linux (dosemu).

I want to rewrite the emulator in C language and adapted to Linux, to add emulation of the Pentagon and the Scorpion, support for TZX format, and some things else.

Nowadays, I have the following computers, some bought on eBay, others in markets:

- Amstrad CPC 464
- Apple iMac G3 333 Mhz
- Atari 1040 STE
- Atari 2600
- Cambridge Z88
- Inves Spectrum +
- MSX Philips VG-8020
- Palm Zire
- Palm Zire 21
- Sinclair QL
- ZX Spectrum 48k

- 2 ZX Spectrum +
- ZX Spectrum +2A
- ZX-81

2 Greetings

I want to thank the help and the collaboration of the following people:

- Ivan Daunis, for the PCGPE and the INTERVUE
- Ruben Parra, for the Spectrum +2A
- To my friends at Mexico
- To my parents and brothers
- To Amstrad, for permitting to use the ROMS to someone who makes an emulator
- Samir Ribic, for helping me with the uploading, and for all the information included in his Warajevo emulator.
- Martijn van der Heide, for installing the emulator in the FTP
- Phillip Kendall, for telling me a bug with the BIT instruction
- Pedro Gimeno, for the timings of some instructions.
- Rodolfo Edison Guerra, for including my emulator in his SpecBase.
- Anyone who makes demos for the Spectrum; they are a good field of testing for my emulator.

- And mainly to this big genius, Clive Sinclair.

3 Contact with the author

If you want to make suggestions or tell me the bugs that the emulator has, you can contact me by the following e-mail:

chernandezba@hotmail.com

I have also made other utilities related to the Spectrum:

- Converter of Spectrum tapes to .TAP files: similar to SMPATAP, but reading directly from tape and working in LINUX.
- Converter of ZX-81 tapes to .P files, readable from the ZX-81 emulator XTENDER by Carlo Delhez.

4 The Emulator

4.1 Distributed files

* Files needed to emulate the Spectrum:

ZXSPECTR.COM	The emulator
48.ROM	It's the Spectrum 16k/48k ROM
INVES.ROM	Inves Spectrum+ ROM
128.ROM	Spectrum 128k ROM
P2.ROM	Spectrum Plus 2 ROM
P2S.ROM	Spectrum Plus 2 ROM (Spanish)
P2F.ROM	Spectrum Plus 2 ROM (French)
P2A40.ROM	Spectrum Plus 2A ROM (Version 4.0)
P2A41.ROM	Spectrum Plus 2A ROM (Version 4.1)
P2AS.ROM	Spectrum Plus 2A ROM (Spanish)
ZXSPECTR.SCR	Introductory screen

* Documentation:

ZXSPECTR.PDF	The file you are reading
ZXSP_ESP.PDF	The same file but in Spanish
NEWS_ESP.TXT	New features and bugs of all versions, in Spanish
NEWS.TXT	New features and bugs of all versions, in English

* Utilities:

LINEASMP.EXE	Program to read sound from tape
SMPATAP.EXE	Program to convert sound to .TAP files
TAPABIN.EXE	Program to convert data from .TAP files to .BIN files
SP_Z80.EXE	Program to convert between SP and Z80 formats (only 48k)
VGA2RAW, VGA2RAW.EXE	Program to convert the VGA output video file to a BGR 24 bit raw video file

* Programs and games:

SPED52.TAP	Assembler/disassembler for 128k. It also includes the source code in his own format (SPED)
CONVERSO.TAP	Converter of source code between formats GENS, TED and SPED. It also includes the source code
REALDEBU.TAP	Disassemblers for 48k and 128k, with source code

CURSORDR.TAP	Painting program. It also includes the source code
ROCMAN.TAP	Game ROCMAN by Xavi Martin Puche.
TOI.TAP	The 4 stages (and the intros) of TOI ACID GAME
SIRFRED.TAP	Game Sir Fred by Made In Spain
RICK.TAP	Rick Dangerous
CANCIONE.TAP	Program in BASIC with songs in PLAY format
CD.TAP	Program that simulates a CD, with many songs. It's one of the first programs of Public Domain for the Spectrum
HISTERIA.TAP	Game Histeria
BUBBLE.TAP	Game BUBBLE BOBBLE
BINARY_L.TAP, NOUMENON.TAP, THECUBE.TAP, POWER_UP.TAP	4 Amazing demos for 128k
TOICLAVE.ZX	Codes for TOI ACID GAME
ABADIA.ZX	Game Abadía del Crimen (128k)
XENO.ZX	Game XENO

4.2 Requirements to run the emulator

The minimum to run the emulator is:

CPU Intel 8088, MS-DOS, 462 Kb RAM free (in its maximum version) + the size of file ZXSPECTR.COM, and VGA video card. Optionally, for a faster emulation of the 128k of Spectrum, you must have EMS 3.0 memory or superior, a Sound Blaster Pro or compatible sound card for emulating the AY chip and for loading from tape. Without Sound Blaster, you can substract 32 Kb to the total amount. And with EMS memory, 256 Kb minus.

The emulator may run with a CPU 8088, because it doesn't use 386 instructions, but a CPU running at 133 MHz is needed in order to have the same speed as a real Spectrum.

You must run it on a DOS session, not on a Windows box; if not, synchronization is not very accurate.

I've even tested the emulator inside MS-DOS emulators for Linux, like the dosemu and the dosbox.

4.3 ZXSPECTR.COM command line

Before you run the emulator, you can specify options and a file name (.SP or .ZX) to load. The options are:

/?	Show the help screen
/Red	Force screen mode with Red tone
/Green	Force screen mode with Green tone
/Blue	Force screen mode with Blue tone

These three options may be combined for running the emulator in different colours, for example /Green/Red to run with yellow, or all the three options to run with gray scale.

/Nosb	Do not use Sound Blaster card
/No386	Do not detect 386 processor
/Noems	Do not use expanded memory
/Eng	View all the menu messages in English
/Esp	View all the menu messages in Spanish

/16k	Emulate Spectrum 16k
/48k	Emulate Spectrum 48k
/Inves	Emulate Inves Spectrum+
/128k	Emulate Spectrum 128k
/P2	Emulate Spectrum Plus 2
/P2F	Emulate Spectrum Plus 2 (French)
/P2S	Emulate Spectrum Plus 2 (Spanish)
/P2A40	Emulate Spectrum Plus 2A (ROM v4.0)
/P2A41	Emulate Spectrum Plus 2A (ROM v4.1)
/P2AS	Emulate Spectrum Plus 2A (Spanish)

4.4 Spectrum emulation

The emulated Spectrum machines are:
(Group 48k)
Sinclair 16k (Partial emulation, only when writing)
Sinclair 48k
Inves Spectrum+

(Group 128k) - These 4 are almost the same, but they have different ROM and external appearance:
Sinclair 128k
Amstrad +2
Amstrad +2 - ROM in French
Amstrad +2 - ROM in Spanish

(Group +2A)
Amstrad +2A (ROM v4.0)
Amstrad +2A (ROM v4.1)
Amstrad +2A - ROM in Spanish

4.4.1 Z80 CPU

CPU emulation is synchronized, but it doesn't emulate the contended memory. Nevertheless, you may enable the pseudo-emulation of the contended memory, far from being real but useful in many

cases: when an instruction runs in the contended memory (16384-32767 in 48k mode and RAMS 4,5,6,7 in 128k mode) or an IN or OUT is made to an even port, that instruction lasts 15% more that its real time. Also, in the menu, there's an option to change the relative speed of the emulator.

You may synchronize the Spectrum CPU by the following ways: using the Timer, I mean, the CPU internal counter, or using the SoundBlaster. So, you have the three following combinations:

- Using the Timer: In this case, an interruption is generated 50 per second. When a complete frame of the Spectrum is drawn on the screen, the interrupt is checked. If it hasn't arrived yet, we wait for it. It's the most accurated method to the Spectrum speed, but the sound is not synchronized with the screen; the sound buffer is assigned by the Timer, and the SoundBlaster timings are not the same as the Timer, in a few seconds we may listen some repeated sounds, and also we may miss another sound fragments.
- Using the Sound Blaster: If the sound card is available, an interrupt is produced every time the sound card empties its sound buffer. In the emulator, the buffer is 10 times a video frame (0.2 seconds). When 10 frames are drawn, we wait for that interrupt.
- Using both the Timer and the Sound Blaster: In this case, every video frame is synchronized with the Timer. Every 10 frames, the last is synchronized with the Sound Blaster. We may listen the sound exactly synchronized with this method.

Every method is recommended in the following cases:

- If you don't want to listen to music, use only the Timer
- If you do want to hear music:
  - In MS-DOS, Windows 95 and Windows 98 environments: We have MS-DOS environment in the three cases, for the most sound cards, it is recommended to use Timer+SoundBlaster
  - In Windows 2000 and XP sessions: SoundBlaster method can be the preferred one, music is not desynchronized, but you may see “steps” on the screen (5 steps every second), I mean, if there's a moving object, you may see it accelerating and decelerating every 5 times by second. If you don't hear well the music, use Timer or Timer+SoundBlaster.

The Timer method is always available; the other two, depends on the Sound Blaster (and the /nosb switch), even if you disable the sound card output or the AY chip is not present (48k modes...).

It emulates the undocumented instructions, like HX, LX, HY, LY registers, opcodes SLL, the 8 RETS with ED prefix: RETN(ED45), RETI(ED4D), RET3(ED55), RET4(ED5D), RET5(ED65), RET6(ED6D), RET7(ED75), RET8(ED7D), and the repeated NEG and IM. Also the OUT (C),F (ED71) and the IN F,(C) (ED70).

It also emulates bit manipulation instructions of kind (XY+d) (DD or FD + CB) with 1 more argument, I mean, it executes the instruction and the result is stored in (XY+d) and in the register indicated, for example: SET 3,(IX+0),C ->It sets bit 3 of (IX+0) and the result of (IX+0) is stored in C.

It also emulates the R register, which is increased by 1 after each instruction, where preffixes are to be regarded as separate instructions; but there's an exception, instructions with preffix DDCB and FDCB increase R by two. R register is an 8 bit register, but when it is increased only the lowest 7 bits are used, and the highest bit remains unchanged (except with the instruction LD R,A).

Bits 3 and 5 of the flags register are emulated partially: they remain constant after a POP AF or an EX AF,AF'. Sign bit and P/V bit have their correct value after a BIT instruction.

It also emulates the interrupt modes of the Spectrum: IM0 and IM1, that call to address 38H (56), and the IM2 mode. They are emulated at a frequency of 50Hz. You can also make a NMI interruption, calling at address 66H (102).

I must tell that Z80 emulation isn't perfect, it has still bugs that I don't know which they are.

4.4.2 The keyboard

The keyboard emulated is the Spectrum 48k one, I mean, 40 keys. Keys ALT and CTRL are both the Symbol Shift. Cursor keys and Ins emulate the Joystick Kempston, and ESC makes appear the emulator menu. It also emulates the following extended keys: Backspace, Caps Lock and TAB (meaning Extended Mode).

It emulates the two kinds of keyboard: Issue 1 and Issue 2 (bit 6 of keyboard port set or reset).

4.4.3 The Joystick Kempston

The joystick kempston is emulated through cursor keys and Ins for fire (it also works with cursors and Ins of the numeric keypad). Any port with bit 5=0 identifies the joystick kempston. It also emulates an autofire of variable speed, found in those old joysticks for Spectrum.

4.4.4 Sound

Sound in Spectrum 48k is generated through port 254, with bits 3 and 4. Bit 3 is only used when saving from BASIC, and bit 4 is used normally. Bit 3 is emulated by a few emulators, but ZXSpectr does emulate it, and it's due to the emulation of the Inves Spectrum+ (for more information about Inves, read the 4.4.10 section).

In the Spectrum, any value sent to bits 3 and 4 different from the last ones sent will produce sound, and ZXSpectr also works in this way. Sound is sent to the Sound Blaster.

It also emulates the sound chip of Spectrum 128 (AY-3-8912), using the Sound Blaster card. It also emulates the stereo sound called ACB/ABC, I mean, one sound channel is heard from left speaker, the other from right speaker, and the third channel from both speakers. In the ACB mode, channel A sounds from left speaker, B from right speaker and C from the centre. In ABC mode, A goes to left speaker, C to right speaker and B to centre. I have only found a few demos that use stereo sound.

Sound Chip is controlled by two ports, which have bit 15 of the address port set and bit 1 reset (A15=1,A1=0). If bit 14 is set (usually port 65533) is the port of register selection (output) or shows the value of the selected register (input), and if bit 14 is 0 (usually port 49149) is the port used to send the value to the register (output only). I say usually they are port 65533 and 49149 because these are the ports mentioned in the manuals, but I have found many games and demos that use other ports.

The registers of the AY chip are:

R0 Fine control of channel A frequency
R1 Coarse control of channel A frequency
R2 Fine control of channel B frequency
R3 Coarse control of channel B frequency
R4 Fine control of channel C frequency
R5 Coarse control of channel C frequency

Frequency is a value of 12 bits formed with bits D3-D0 of coarse control register and bits D7-D0 of fine control register. Basic unity of frequency is the clock frequency divided by 16 (110.83 Khz). Frequencies are between 27 Hz and 110 KHz.

R6 Noise Control D4-D0
Noise period is made with lower 5 bits divided by 16. To emulate noise, it sends every time period a 0 or 1 randomly; to generate this value, we use the same RND Spectrum function: n=(75*(n+1)-1)/65536

R7 Mixer control
D0 if D0=0, channel A produce tone
D1 if D1=0, channel B produce tone
D2 if D2=0, channel C produce tone
D3 if D3=0, channel A produce noise
D4 if D4=0, channel B produce noise
D5 if D5=0, channel C produce noise
D6 if D6=0, register R14 is input port, else it is output port
D7 not used

R8 Channel A volume
R9 Channel B volume
RA Channel C volume
D4 1=use envelope generator (not emulated, volume is constant if D4=1)
0=use the value in D3-D0 as a volume
D3-D0 Volume

RB Fine control of envelope period
RC Coarse control of envelope period
These two registers are not emulated

RD Envelope control register (not emulated)
D3 CONTINUE bit
D2 ATTACK bit
D1 ALTERNATE bit
D0 HOLD bit

RE Controls RS-232, KeyPad, and MIDI (not emulated)
RF Not used

4.4.5 Screen

At the screen you have the 16 colours of the Spectrum and real flash (exchange between PAPER and INK every 16 frames). You may also control the brightness of colours.

Screen updating is made every video frame (50 times per second). It has autoframeskip, I mean, if it lasts more than 1/50 second per frame, it won't be drawn on the screen.If there's not time even to display one frame on the last second, an screen update will be forced.

The emulator also has a screen saver; I don't want to tell you how it is, I prefer you see it. It appears when no key is pressed during 2 minutes.

4.4.6 128k of memory

128k can be emulated with expanded memory (EMS) or with RAM. With EMS, paging is fast; however, with RAM, paging may be very slow, running a bit faster if you have a 386 or superior. The emulator autodetects if the CPU is a 386 or superior; I have tested it on a XT and a Pentium, but not on a 286. If the emulator crashes when it is executed and you have a 286 or 186, use the /no386 option.

The memory needed to run the emulator is:

- The size of the file ZXSPECTR.COM
- Plus 188 Kb RAM free
- Plus 45 Kb RAM free if SoundBlaster used
- Plus 256 Kb RAM free if you don't have expanded memory (EMS)

Paging of the 128k is made with port 32765 (actually is any port with A1=0 and A15=0). Description of the port 32765 is:

D0 a D2	RAM selection
D3	Screen selection (screen on RAM 5 or RAM 7)
D4	ROM Selection
D5	Paging disable

Due to an internal error in the 128k (but not in a +2A), if we read that port, we will actually OUT to this port with value 255, crashing the computer if we are in 128 BASIC. This feature is also emulated.

Paging of the +2A is made with 2 ports: 32765 (and only the 32765) and 8189.

Description of port 8189 is:

D0	Make D1 y D2 control ROM or RAM [1]

1 In the book says bit D3

D1 y D2 ROM/RAM paging [2]
D4 Disk Motor (not emulated)
D5 STROBE signal in parallel port (active with 0) (not emulated)

When bit 0 of port 8189 is 0, ROM is selected with bit 4 of 32765 (low bit of ROM) and bit 2 of 8189 (high bit of ROM):

Bit 2 of 8189	Bit 4 of 32765	ROM switched
0	0	0
0	1	1
1	0	2
1	1	3

When bit 0 of port 8189 is 1, bits 1 and 2 control which RAM pages use the 64k:

Bit 2 of 8189	Bit 1 of 8189	RAM pages switched
0	0	0,1,2,3
0	1	4,5,6,7
1	0	4,5,6,3
1	1	4,7,6,3

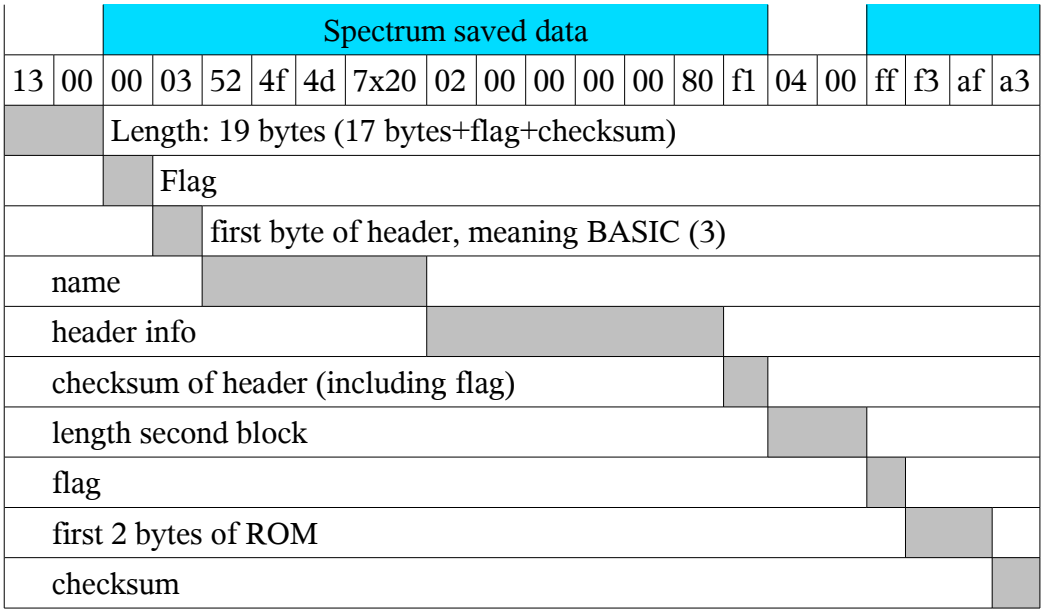
4.4.7 Tape emulation

Tape emulation inside the emulator is made with .TAP files. These files were created in the Z80 emulator of Gerton Lunter. TAP files contains many blocks which identify the tape data saved. Each block has the format:

- WORD that indicates the length of the following data (including flag and checksum).
- BYTE that indicates the flag of the saved block (like a real tape: 0 for headers and 255 for data).
- DATA: Data saved.
- BYTE that indicates the checksum of the data (including flag). It is the result of make XOR with all data.

Example:
SAVE "ROM" CODE 0,2
File .TAP generated:

2 In the book says bit D0 and D1



These files are used inside the emulator when a LOAD or SAVE operation is made. You must specify the .TAP files to use inside the menu.

LOAD and SAVE routines start in ROM at address 1366 and 1218. If these routines were executed, we would only see border flashes. To use TAP files, the emulator knows when where are running any of these ROM routines and it will be forwarded to custom .TAP tape emulation functions. It is allowed to load data shorter than the contained in the .TAP file, using the following byte as checksum.

Block verifying is not permitted (the emulator always loads the block).

Flag Z' is set to 0 in the ROM routine, but you can enter at address 1378 setting flag Z' to 1, and it says the Spectrum it must not distinguish the flag, loading it as a normal byte. It works in this way on a real Spectrum and on the emulator. It's used on copiers and some games; however, I have found problems with some games, like ROCMAN, and you can disable the option of allowing any-flag-loading from the menu (flag Z' always go 0). I have tested this game with Z80 emulator (3.05) and it crashes; in Warajevo (2.51) it doesn't crash, because it doesn't allow any-flag-loading (in fast loading mode).

4.4.8 Real tape loading

The emulator has an option at the menu to read a program from a real tape and convert it to TAP file. You need a compatible Sound Blaster Pro card or superior. Also, you need a tape recorder (obviously!) and a mono cable to connect it to the tape recorder and to Line In of Sound Blaster. You can't read in a DOS Window, it must be done on a DOS session. I also recommend not to have any disk cache in memory, like SMARTDRV.

Inside the emulator you can change the reading speed, for loading turbo tapes (more than 1500

bauds); however, there are some SB cards that doesn't allow to change it. If you can't change the speed, you may change the loading constants to read turbo tapes. These constants are described in paragraph 5. Menu options.

4.4.9 Snapshot Files

The first versions of the emulator worked only with SP files, from the SPECTRUM emulator by Pedro Gimeno. Then, I created my own file format, called ZX, which is derived from SP format but with more extra data (ZX header is like SP but with 256 bytes more). I know there's an emulator which uses files with extension ZX, but they aren't related with my ZX files.

New versions of the emulator (since version 1.1) can load SP or ZX files, but saving is only made with ZX files. The description of SP format is the following (I have copied it from SPECTRUM.DOC without changing anything):

Offset	Longitud	Descripción
0	2 bytes	"SP" (53h, 50h) Signatura
2	1 palabra	Longitud del programa en bytes (el emulador actualmente sólo genera programas de 49152 bytes)
4	1 palabra	Posición inicial del programa (el emulador actualmente sólo genera programas que comiencen en la pos. 16384)
6	1 palabra	Registro BC del Z80
8	1 palabra	Registro DE del Z80
10	1 palabra	Registro HL del Z80
12	1 palabra	Registro AF del Z80
14	1 palabra	Registro IX del Z80
16	1 palabra	Registro IY del Z80
18	1 palabra	Registro BC' del Z80
20	1 palabra	Registro DE' del Z80
22	1 palabra	Registro HL' del Z80
24	1 palabra	Registro AF' del Z80
26	1 byte	Registro R (de refresco) del Z80
27	1 byte	Registro I (de interrupciones) del Z80
28	1 palabra	Registro SP del Z80
30	1 palabra	Registro PC del Z80
32	1 palabra	Reservada para uso futuro, siempre 0
34	1 byte	Color del borde al comenzar

<table>
<tr><th>Offset</th><th>Longitud</th><th>Descripción</th></tr>
<tr><td>35</td><td>1 byte</td><td>Reservado para uso futuro, siempre 0</td></tr>
<tr><td>36</td><td>1 palabra</td><td>Palabra de estado codificada por bits.
Formato:
<table>
<tr><th>Bit</th><th>Descripción</th></tr>
<tr><td>15-8</td><td>Reservados para uso futuro</td></tr>
<tr><td>7-6</td><td>Reservados para uso interno, siempre 0</td></tr>
<tr><td>5</td><td>Estado del Flash:
0 - tinta INK, papel PAPER
1 - tinta PAPER, papel INK</td></tr>
<tr><td>4</td><td>Interrupción pendiente de ejecutarse</td></tr>
<tr><td>3</td><td>Reservado para uso futuro</td></tr>
<tr><td>2</td><td>Biestable IFF2 (uso interno)</td></tr>
<tr><td>1</td><td>Modo de interrupción: 0=IM1; 1=IM2</td></tr>
<tr><td>0</td><td>Biestable IFF1 (estado de interrupción):
0 - Interrupciones desactivadas (DI)
1 - Interrupciones activadas (EI)</td></tr>
</table></td></tr>
</table>

ZX format has 3 versions, version 1 is the first one and version 3 is the one used now. Values used from version 2 are indicated with (v. 2+). Version 3 only have one more value, the Machine emulated.

The description for ZX files is:

Field	Description
signatura	"ZX"
long_prog	WORD=49152
pos_inicial	WORD=16384
reg_c	BYTE
reg_b	BYTE
reg_e	BYTE
reg_d	BYTE
reg_l	BYTE
reg_h	BYTE
reg_f	BYTE
reg_a	BYTE

<table>
<tr><td>Field</td><td>Description</td></tr>
<tr><td>reg_ixl</td><td>BYTE</td></tr>
<tr><td>reg_ixh</td><td>BYTE</td></tr>
<tr><td>reg_iyl</td><td>BYTE</td></tr>
<tr><td>reg_iyh</td><td>BYTE</td></tr>
<tr><td colspan="2">Registers '</td></tr>
<tr><td>reg_c_</td><td>BYTE</td></tr>
<tr><td>reg_b_</td><td>BYTE</td></tr>
<tr><td>reg_e_</td><td>BYTE</td></tr>
<tr><td>reg_d_</td><td>BYTE</td></tr>
<tr><td>reg_l_</td><td>BYTE</td></tr>
<tr><td>reg_h_</td><td>BYTE</td></tr>
<tr><td>reg_f_</td><td>BYTE</td></tr>
<tr><td>reg_a_</td><td>BYTE</td></tr>
<tr><td>reg_r</td><td>BYTE</td></tr>
<tr><td>reg_i</td><td>BYTE</td></tr>
<tr><td>reg_sp</td><td>WORD</td></tr>
<tr><td>reg_pc</td><td>WORD</td></tr>
<tr><td>reservado1</td><td>WORD=0</td></tr>
<tr><td>border</td><td>BYTE</td></tr>
<tr><td>reservado2</td><td>BYTE=0 In version 1 of header, it was a WORD an reservado20 didn't exist.</td></tr>
<tr><td>bits_estado</td><td><table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>7-6</td><td>Reserved for internal use, always 0</td></tr>
<tr><td>5</td><td>Flash state: 0 - ink INK, paper PAPER. 1 - ink PAPER, paper INK</td></tr>
<tr><td>4</td><td>Interruption pending</td></tr>
<tr><td>3</td><td>Reserved for internal use</td></tr>
<tr><td>2</td><td>Flip-flop IFF2 (internal use)</td></tr>
<tr><td>1</td><td>Interrupt mode: 0=IM1, 1=IM2</td></tr>
<tr><td>0</td><td>Flip-flop IFF1 (interrupt state):
0 – Interrupts disabled (DI)
1 – Interrupts enabled (EI)
The emulator only uses bit 5 (flash state), bit 1 (mode IM0/IM1 or IM2) and bit 0 (DI or EI)</td></tr>
</table></td></tr>
<tr><td>reservado20</td><td>BYTE=0 (v. 2+)</td></tr>
<tr><td colspan="2">Next bytes are the extended header of ZX files</td></tr>
<tr><td>version</td><td>BYTE=1, 2 o 3 ZX Header version</td></tr>
</table>

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>cabecera_velocidad</td><td>WORD Not used since ZXSpectr V2.0</td></tr>
<tr><td>cabecera_dirs_pant</td><td>WORD=64 Not used. Used in the CGA version and screen updating. Do not use in next versions</td></tr>
<tr><td>cabecera_frecuencia</td><td>BYTE=1 Not used. Used in the CGA version and screen updating. Do not use in next versions</td></tr>
<tr><td>control_brillo</td><td>BYTE Brightness value</td></tr>
<tr><td>disparador_defecto</td><td>BYTE Autofire frequency. Calculated by frecuencia = 50/disparador_defecto</td></tr>
<tr><td>sonido</td><td>BYTE 1 if sound is enabled (port 254)</td></tr>
<tr><td>bits_estado0</td><td>BYTE The meaning is the following:

0

<table><tr><th>Bit</th><th>Description</th></tr><tr><td>7</td><td>Contains bit 6 of keyboard port (Issue 1 or 2)</td></tr><tr><td>6</td><td>1 if auto-fire enabled</td></tr><tr><td>5</td><td>0 if Flash refresh is enabled. Obsolete</td></tr><tr><td>4</td><td>1 if snapshot is 128k (version 2+)</td></tr><tr><td>3</td><td>To 1 if border changing is not possible using OUT 211, value (version 2+)</td></tr><tr><td>2</td><td>Not used</td></tr><tr><td>1</td><td>1 if screen saver is enabled</td></tr><tr><td>0</td><td>CGA Palette (v. 2+)
0=Black,Green,Red,Yellow
1=Cyan,Magenta,White</td></tr></table></td></tr>
<tr><td>puerto_32765</td><td>BYTE Value of last OUT to port 32765 (v. 2+)</td></tr>
<tr><td>puerto_8189</td><td>BYTE Value of last OUT to port 8189 (v. 2+)</td></tr>
<tr><td>paginas_actuales</td><td>4 BYTES Contains for each memory segment (0000h-3fffh, 4000h-7fffh, 8000h-bfffh, c000h-ffffh) the page (ROM o RAM) assigned, meaning from 0 to 3 ROMS, and from 4 to 11 RAMS (v. 2+). If the machine emulated is a 128k (but not a +2A), ROM1 is identified as ROM3.</td></tr>
<tr><td>puerto_65533</td><td>BYTE Value of last OUT to port 65533 (v. 2+)</td></tr>
<tr><td>ay_3_8912_registros</td><td>16 BYTES Values of AY sound chip registers (v. 2+). It is used since an advanced version 2. If it's read in an earlier version, we will see 255, because it was reserved.</td></tr>
</table>

Field	Description
ordenador_emulado	BYTE Machine emulated: (v. 3+) 0=Sinclair 16k 1=Sinclair 48k 2=Inves Spectrum+ 3=Sinclair 128k 4=Amstrad +2 5=Amstrad +2 - French 6=Amstrad +2 - Spanish 7=Amstrad +2A (ROM v4.0) 8=Amstrad +2A (ROM v4.1) 9=Amstrad +2A - Spanish
reservado4	222 BYTES=255 Reserved for future use
DATOS	X BYTES

From “DATOS” memory bytes are saved. If the program is 48k, data is saved from address 16384 until 65535; if it's 128k, data are saved from RAM 0 until RAM 7. The emulator generates a 48k file if it runs in /48k mode, or if it's in 48 BASIC mode, else 128k are saved.

Memory bytes are compressed, in the following manner:

- If a byte is repeated more than 4 times, it is saved like:

221,221,byte to repeat,times repeated

- If there's a 221 byte just before the repetition, it is saved as:

221 before repetition,byte to repeat,221,221,byte to repeat,times repeated-1

When 128k are saved, it is made in two blocks of 64k, the first block from RAM 0 until RAM 3, and the second one from RAM 4 until RAM 7, and bytes repeated which last from RAM 3 to RAM 4 are not detected, for example:

If we have the following:
RAM3
16377=10
16378=20
16379=20
16380=20
16381=20
16382=20
16383=20
RAM4
0 =20
1 =20
2 =20
3 =20
4 =20
5 =20
6 =20
7 =4

The data saved is:
10, 221,221,20,6, 221,221,20,7, 4

4.4.10 Inves Spectrum+ Emulation

The Inves Spectrum+ was a computer created by Investronica near 1988. It has some bugs compared to a Spectrum:

The ROM is similar to the Spectrum 128k ROM 1, although it has only 48k RAM, and it has the paging routines at the address 14446.

Screen updating is different from a Spectrum, in the manner that when an interruption is produced, the ULA starts drawing at the screen zone (address 16384), but not at the top of the border, So, in some games that use the time of the top border to erase objects, in the Inves we will see flickering. Also, in many game screens we may see "false" lines when a combination of colours is produced (I don't know yet which combination produce this effect). I don't know the Inves timings, but I suppose they are the same as the Spectrum ones.

The most interesting feature of the Inves Spectrum+ is the effect produced when POKEing in the ROM, yes, in the ROM. There are some "privileged" adresses of the ROM that may alterate the normal operation of the Inves; these addresses have the low byte with 254 (XXFEh), like the speaker port.

Pokeing in all these addresses of the ROM (254,511,,...,16383) with the same value, we will make an AND mask that will be applied to the value sent to port 254, and the resulting value will be the real value sent. We must assume that the POKE value in ROM when switching the Inves on (but not when making RESET) is 255. Thus, we will produce two effects:

1. First is that the three low bits are a mask applied to the border when we change it, I mean, the value sent to port 254 is ANDed with the ROM POKE value, and the resulting value will be the colour of the border: for example: with a ROM POKE value of 6, we will get Borders with even number, I mean, border 0 and 1 will be 0, 2 and 3 will be 2, etc. On the other hand, if we POKE with 0, all the borders will be black (0).
2. The second effect (and the worst in a lot of games with music) affects the sound. I'll try to explain it on an easy way, but it's a bit complicated. Bits 3 and 4 of the value sent to port 254 are ANDed with the ROM POKE value. Then, the resulting bits 3 and 4 are XORed and we get 1 bit of result. Then, this bit is sent to the speaker, in the manner that if we sent a bit equal to the previous bit the speaker doesn't swicth and we hear nothing. We must remember that in a normal Spectrum, any value sent to bits 3 and 4 different from the previous ones produce sound. So, we have the following combinations: (Y means that a sound is produced, N means no sound)

	Bits 4 and 3 ROM POKE value			
Bits 4 and 3 port 254 (first and second value sent)	**11**	**10**	**01**	**00**
00 **01**	Y	N	Y	N

	Bits 4 and 3 ROM POKE value			
00 10	Y	Y	N	N
00 11	N	Y	Y	N
01 10	N	Y	Y	N
01 11	Y	Y	N	N
10 11	Y	N	Y	N

In a normal Spectrum, any combination of these bits 4 and 3 in the port 254 will produce sound; in an Inves (and in the emulator) it is not the same. In a real Inves, we must POKE all the ROM addresses which have the low byte equal to 254 (XXFEH); if we only POKE some of them with the same value, the resulting effects will be a combination of all (there's a combination that, when we make a BEEP, we will see black lines on the border!). In the emulator, we must only POKE one of these addresses (we may also do it in the menu).

The effect of this sound feature of the Inves makes that we can't hear the music in many games, or we only hear the rhythm in two-channel music (all the Code Masters games, the Lemmings,....). Also, we can change the sound operation making a POKE 23659,0 from BASIC. As you may see in the table, there's not any ROM POKE value that makes the Inves work as a Spectrum.

4.4.11 Bus idle port

A "bus idle" port is called any Spectrum port which doesn't have any device assigned. That port returns the value that the ULA reads from the screen, I mean, when the ULA is drawing the screen it reads bytes from the pixels and the attributes. This byte read by the ULA may be known reading (IN) any port with no device attached (usually port 255). When the ULA doesn't read bytes from the screen (when drawing the border or while the synchronization) a value of 255 is returned. It works in the Spectrum 16k, 48k, and 128k (and +2). But, in an Inves Spectrum+ we will always read 255.

In an Spectrum +2A, it's a bit more complicated:

When the paging is disabled (bit 5 of port 32765 set to 1), we will always read 255; when it is not, we will have the byte that the ULA reads (with bit 0 set) if the port number has the following mask: 0000XXXXXXXXXX01b, I mean, starting from port 1, stepping 4, until port 4093 (1,5,9,13,..., 4093). I have discovered this feature making some tests with my +2A, and all the documentation and FAQS from emulators and Web pages say that the +2A always returns 255. Maybe the mistake is to read only port 255, that it does return 255.

This feature of the "bus idle" port is used in some games, like Arkanoid and Renegade. Starting from version 3.0 of the emulator it works quite good.

5 Menu options

To choose a menu option you may use the arrow keys and Enter, + and -. In the case of options with numeric values, you must use + to increment and - to decrement the value.

There are some options that can't be chosen; in this case, the normal cursor (->) will be (-x).

In options load and save, insert and extract, we may choose the file pressing Enter and using the arrow keys and PgUp and PgDn. If we want to write the whole filename, we must choose the option with the key - or Space.

5.1 Load Snapshot

With this option we load an snapshot file (ZX or SP); the format of the file is detected from its header, and not from its extension, for example: you may have a file with extension .TXT and be a ZX file. When you load a SP file, options of brightness, autofire, etc, will be the ones used before the loading.

5.2 Save Snapshot

With this option we make a ZX snapshot file; you must write name and extension. If the file already exists, you will be asked to replace it.

5.3 Machine Selection

With this option we will choose the computer to emulate. When selected, a RESET is done, relative speed is set to 100% and tapes are extracted.

5.4 Screen Settings

5.4.1 Load Screen

With this option we will load a .SCR file in address 16384.

5.4.2 Save Screen

With this option we will save the addresses 16384-23295 in a .SCR file (we must write the

extension).

5.4.3 Brightness control

With this option we can increase or decrease the brightness of colours; menu colours aren't changed.

5.4.4 Screen saver

With this option we can disable the screen saver; sometimes is convenient to disable it, specially when we are hearing the music of a game.

5.4.5 Start recording video

With this option we start the recording of the screen output to a file. It's a raw file, without header or compression. It contains the VGA screen output for each recorded frame (each frame it's 64kb length). The VGA mode is a colour palette mode, so we must convert the file using the utility VGA2RAW. Then, the final file it's a RGB colour file. I mean: 320x200, BGR24 (24 bit each screen pixel).

We can play the file using the program Mplayer, for example:

mplayer -demuxer rawvideo -rawvideo fps=2:w=320:h=200:format=bgr24 file.rwv

For a video file with 2 FPS.

And if we want to play also raw audio (see section 5.8.6 Start recording audio):

mplayer -demuxer rawvideo -rawvideo fps=2:w=320:h=200:format=bgr24 file.rwv -audiofile file.rwa -audio-demuxer 20 -rawaudio channels=2:rate=15550:samplesize=1

We must know that the video recording is made when we are in multitask mode or when we exit the menu and return to the emulator, like the audio recording. Although when we are recording video the audio sounds bad, the output file will always be recorded in excellent conditions.

5.4.6 File

With this option we select the video output file. It is preferred the .VGA extension when using the file selector.

5.4.7 FPS

With this option we tell how many frames per second use to record the file.

5.5 Debug menu

5.5.1 Generate RESET

It makes a RESET of the Spectrum.

5.5.2 Generate NMI

It makes a non-maskable interrupt (call to address 66H).

5.5.3 Show registers

With this option we can see the Z80 registers. You may also enable or disable the interruptions and change the interrupt mode.

We may also see the active pages, the active screen and whether the paging is enabled or not (in 128k). Also the value of port 254 (border colour and bits 4 and 3) and the number of screen frames per second (FPS).

5.5.4 Poke

It is used to change memory bytes, useful to get infinite lives in games. The previous value of the address is shown.

5.6 Tape emulation

5.6.1 Insert/Eject Input file

With this option we tell the emulator the file to use (of TAP format) as an input tape (if we select insert), or that the file is not used anymore (if we select eject). If the file doesn't exist, you will be asked to create it.

5.6.2 File

In this option we tell the emulator the file to use as an input tape (of TAP format). We also must insert the file in order the emulator use it.

5.6.3 Insert/Eject Output file

With this option we tell the emulator the file to use (of TAP format) as an output tape (if we select insert), or that the file is not used anymore (if we select eject). If the file doesn't exist, you will be asked to create it.

5.6.4 File

In this option we tell the emulator the file to use as an output tape (of TAP format). We also must insert the file in order the emulator use it.

5.6.5 Any flag loading

Here we tell the ROM LOAD routine to allow load data independently of its flag (Z' set), or if it is not allowed (Z' always 0).

5.6.6 Load From Tape

This is the menu to load programs from a real tape of Spectrum. You must have a Sound Blaster or compatible card to have the option active. Also, you musn't run the option in Windows, if you do it, you will load nothing (what strange!).

5.6.6.1 Bauds

It indicates the loading speed. It is done by changing the reading frequency of the sound card. In my sound card I may change the speed but it always reads at 11KHz. You can also change the loading speed by changing values length of zero wave and one wave.

5.6.6.2 Length of leader signal wave

It shows how lasts one leader signal wave; length (in seconds) of this value is: seconds=length of leader signal wave/frequency. Usually, in turbo loading, leader signal doesn't change.

5.6.6.3 Length of leader signal

It indicates how many waves of leader signal must be found in order to recognize them as a leader signal. When loading, when this value is surpassed, we may see at the screen "Reading ld signal..." (Reading leader signal).

5.6.6.4 Length of zero wave

It indicates how lasts the wave that represents a zero bit, with an error margin of +-2. The length is calculated like in leader signal wave. You may only change it if your sound card doesn't allow to change the frequency: if you have to read a turbo tape at 3000 bauds, length of zero and one waves must be divided by 2.

5.6.6.5 Length of one wave

It indicates how lasts the wave that represents a one bit, with an error margin of +-2. The length is calculated like in leader signal wave.

5.6.6.6 Input filter

It shows the filter that uses the sound card when reading data. I've read that the filter is only available with Sound Blaster Pro cards, and not on newer cards.

5.6.6.7 Left Channel Volume

It indicates the input volume for the left channel.

5.6.6.8 Right Channel Volume

It indicates the input volume for the right channel.

5.6.6.9 Show Border

With this option you can see the border while loading; it is not exactly as the border shown while loading on a real Spectrum.

5.6.6.10 Checksum autocorrection

It tells the emulator if the last byte of loading (checksum) must be recalculated when loading error.

5.6.6.11 Start loading

This is the option that I think the most important, which allows to read programs. Before this, you must have a .TAP file inserted as an output tape.

You must know you can't load on a DOS Window (you must run a DOS session), and I

recommend not to have disk caches (like SMARTDRV).

When we select this option, a volume indicator is shown at the half of the screen. It's not important the volume it shows, because you can load at a high volume or at a low volume, it only tells that sound is being read. You may also hear the sound at the speakers. You may also see the border while loading; it's similar as on a real Spectrum.

For a good loading, sound must be heard sharp (adjusting the screw or "azimut" of tape recorder if needed) and at a volume that doesn't "harm" the sound, like on a real Spectrum.

Under the volume indicator, there's a zone that indicates the bytes read, including the flag. It may be possible that bytes shown "stop" on the screen each 1.5 seconds. It happens when you run the emulator with an EMS emulator, like EMM386. The reason of this delay is that direct reading (DMA) is not done byte by byte, but first the whole buffer (16k) is read and then it is passed to the emulator. Don't worry, data reading doesn't stop, but first passes to the intermediate buffer. Nevertheless, if we run the emulator in MS-DOS without EMS, visualization of data is continuous.

When a block is read, header is shown if block is 17 bytes length and has flag 0, if not, flag and length is shown (this length also indicates checksum byte and flag). It also tells us if there's a loading error.

Then, we may save the data pressing the G key. If there's a loading error and the checksum autocalculate is off, when we make LOAD "" the loading error will be shown from BASIC.

5.7 Hardware settings

5.7.1 Keyboard Issue0/1

It allows to choose between the two kinds of keyboard of the Spectrum.

5.7.2 AutoFire

You can enable or disable the Kempston autofire.

5.7.3 AutoFire frequency

It indicates the autofire frequency. This frequency indicates how many times per second bit 4 of kempston port is inverted.

5.7.4 ROM POKE value

It indicates the ROM POKE value (only in Inves mode).

5.7.5 Contended memory

You may enable the pseudo-emulation of the contended memory, that makes an instruction running in a slow RAM lasts 15% more that its real time. In some games it's very real, but I recommend to disable in demos.

5.7.6 Synchronism

You may select the CPU synchronization mode between Timer, SoundBlaster or SoundBlaster+Timer

5.8 Audio Settings

5.8.1 Output Sound

It indicates where to play sound or not.

5.8.2 AY Chip present

It indicates whether the AY sound chip exists or not. The chip can be present without having a Sound Blaster; in this case, AY ports may be manipulated but you hear nothing.

5.8.3 Stereo Mode

It shows the Stereo mode used for the AY chip: ACB,ABC or Mono.

5.8.4 Noise Emulation

You may disable the emulation of the noise channels.

5.8.5 Show AY Chip registers

With this option we will see the 16 internal registers of the sound chip (if it exists).

5.8.6 Start recording audio

With this option we start the recording of the output generated in the emulator, the beeper and the AY chip. We don't need a Sound Blaster to do it. The resulting file is a raw file, without header or compression. The file format is: stereo, 8 bit signed, 15550 Hz (15600 Hz for 48k modes).

We can play the file using the program Mplayer, for example:

mplayer -demuxer 20 -rawaudio channels=2:rate=15550:samplesize=1 archivo.rwa

For a file recorded in 128k mode.

And if we want to play also raw video (see section 5.4.5 Start recording video):

mplayer -demuxer rawvideo -rawvideo fps=2:w=320:h=200:format=bgr24 archivo.rwv -audiofile archivo.rwa -audio-demuxer 20 -rawaudio channels=2:rate=15550:samplesize=1

We must know that the video recording is made when we are in multitask mode or when we exit the menu and return to the emulator, like the video recording. Although when we are recording audio the audio sounds bad, the output file will always be recorded in excellent conditions.

5.8.7 File

With this option we select the audio output file. It is preferred the .RWA extension when using the file selector.

5.9 Language Selection

Here we may choose the language of the menu (English or Spanish).

5.10 Multitasking

We tell if we want the Z80 emulation be active when we are in the menu.

5.11 CPU Speed

Here we may change the relative speed of the Z80.

5.12 Back to the Emulator

With this option we will return to the Spectrum emulation.

5.13 Exit emulator

This option makes the emulator finish.

6 Included utilities

6.1 LINEASMP.EXE

This programs is used to read sound from the LINE IN of Sound Blaster and save it as a SMP file. This SMP file can be edited with the program Fast Tracker, in order to modify the sound in a loading error.

As you may guess, it doesn't work in Windows, and neither works well with SMARTDRV.

When we run it, Sound Blaster will be detected, and then you may specify the loading speed (in bauds). After this you must tell the file SMP to make, writing the extension too. When we want to stop reading, we must press any key.

The SMP file is saved in mono, 8 bits, and at 11KHz (if it is a turbo loading, you must specify the bauds and the program saves it as the loading speed were 1500 bauds).

6.2 SMPATAP.EXE

This program is used to read data from a SMP file and save it as a TAP file. With this program, I bring the possibility to read programs from computers without Sound Blaster.

When we run the program, SMP file and TAP file must be written (both with extension). When it loads a block, you must press any key except the N to save the block.

The SMP file can be generated with a WAV file and the program Fast Tracker, or directly from Fast Tracker or LINEASMP.EXE. It must be mono, 8 bits and at 11KHz.

6.3 TAPABIN.EXE

This program is used to extract data blocks from a TAP file and save them with BIN format.
When the program starts, you must specify the TAP and BIN files to use (both with extension). When it reads a block, you must press any key except the N to save it. The file saved is the data get from a block without flag and checksum; for example: if we want to save a Spectrum screen in BIN format, we must save the block shown as: "Flag:255 Longitud:6912" and not the block: "Bytes:Screen Longitud:6912 Inicio:16384", if not, we would save the header, I mean, 17 bytes.

6.4 SP_Z80.EXE

This program is used to convert files between SP and Z80 formats. You must specify the source file and the target file (both with extension, always .SP or .Z80). It can only convert 48k files.

6.5 VGA2RAW

This program is used to convert the output video file from the VGA pallette format to a BGR24 raw video format. There are the MS-DOS version and Linux (Intel x86 32-bit).

7 Utilities in TAP format

7.1 SPED52.TAP

This tape includes an assembler/disassembler for ZX Spectrum 128k, with its source code. This assembler (SPED) uses the whole 128k of memory. The SPED uses RAM1, and the object code uses RAM 4; source code uses RAM "48" (5,2,0) and RAMs 6 and 7, having a maximum source code of 74549 bytes (41781+16384+16384).

Sentences ORG and ENT are a bit different from other assemblers. ORG always indicates to which address the object code is generated; normally, we assemble at 49152, which corresponds to RAM 4. ENT indicates to which address labels are referred (in a relocatable program, ENT is often 0).

Operator $ always indicates to which address is referred the line where it is situated (indicated with ENT). Operator !, that I have invented, shows where the object code is being generated (indicated with ORG).

To see the listing while assembling, you must write in the source code the LST+ operator, and to make it disappear, LST- .

The assembler can generate a debug info, that consists in a table with each memory address of object code with its corresponding line of source code. This table uses RAM 7; we may see if 3rd block of source code is used, debug info can't be used. The maximum size of the table is 16k (4096 lines). To tell the assembler create debug info, use the operator DPR+ in the source code (and to disable it, DPR-).

When we go to the disassembler, it there's debug info, the original line of the source code will be shown.

Label table uses RAM 3, having a maximum of 16k for labels (it means 1260 labels).

The editor also uses RAM 3, for copying text in a buffer to make the typical actions of Copy/Paste/Move; the maximum is 16k of text.

I must tell that if you haven't EMS memory, when you assemble you may wait a long long time assembling; you'd better go to MS-DOS with EMS active.

I don't want to explain all the options, because it has some help screens (sorry, in Spanish) (in the main environment and in the disassembler, key H; in the editor, Extended Mode and H). Also, you have the source code to see the whole program.

Only one more thing, the SPED format. The assembler only recognizes source code with headers of SPED format, which have a size of 34 bytes but they can be read from BASIC. The description of the header is:

Offset	*Length*	*Description*
0	1	Byte is 3. Indicates BYTES block
1	10	File Name

Offset	*Length*	*Description*
11	2	Word which indicates the length of the first text block
13	4	4 Bytes not used
17	1	Checksum byte. It is used to cheat to BASIC and tell the header is 17 bytes length; the header can be read in a real Spectrum and in the emulator
18	1	Recording day of source code
19	1	Recording month of source code
20	1	Recording (Year-1980) of source code. It is ready to year 2000!
22	2	Word which indicates the length of the second text block. If there's not, is 0
24	2	Word which indicates the length of the third text block. It there's not, is 0
26	9	9 Bytes to 255, used in future versions

After the header comes the text, formed by 1, 2 or 3 blocks with flag 255.

Text blocks are saved in ASCII format, setting bit 7 of a character to make a tabulator, and code 13 for line end. Of all the assemblers that I have for the Spectrum (SPED, GENS, TED), this assembler uses the shortest text format.

7.2 CONVERSO.TAP

This program is a source code converter of formats GENS, TED and SPED. Possible conversions are: GEN-> TED, GEN->SPED, TED->SPED. Source code is included, in SPED format. Note: TED is a format of an editor/assembler which appeared in the MICROHOBBY magazine, of Hobby Press.

7.3 REALDEBU.TAP

It consists on 2 disassemblers, one for 48k and the other for 128k. They are both like MONS. Both source codes are included, in SPED format. They are both named as (R)EAL DEBUG, that's because they control perfectly the R register, and some other disassemblers, like MONS, don't.

The 48k disassembler is relocatable. To run it:
CLEAR address-1: LOAD "RDEBUG6.CO" CODE address:
RANDOMIZE USR address

To know the keys of the 48k run the 128k disassembler, it has a help screen: there are only two options that the 48k one doesn't have, RAM selection and Assembly.

The 128k disassembler uses RAM 1, plus a block of 256 bytes length placed below address 49152. To run it:

From 128 BASIC: RANDOMIZE USR 0. With this, we go to ROM3. Then:

CLEAR 29999: OUT 32765,17: LOAD "R37DE128.C" CODE 49152: RANDOMIZE address: RANDOMIZE USR 49152

CLEAR can be what you want, but it must be below 49152. The "address" variable indicates where the 256 bytes block is placed; it can be placed, for example, at 23296. If "address" is other than 23296, then the CLEAR must be below "address".

7.4 CURSORDR.TAP

This is a painting program, the CURSOR DRAW, which also includes the source code in SPED format.

And that's all friends!. I hope you enjoy the emulator. :-)

CESAR HERNANDEZ BAÑO
chernandezba@hotmail.com

P.D. Sorry for my bad English.