Minimal PET on stick

Jesús Arias

This is a recreation of a Commodore PET into an Lattice ICESTICK board. But, I must remark this isn't an accurate one because some parts of the original computer were removed in order to fit it into the restricted space of the board's FPGA. Let's first present the components of that computer (model 2001), that are:

- A 6502 processor.
- 4KB or 8KB of RAM. (up to 32KB on other models)
- 14KB of ROM.
- Monochrome, text mode, video controller with:
  - 40×25 character array.
  - 1KB of video RAM.
  - 2KB of ROM for character generation. It includes two sets of 128 characters (PETscii):
    1. "Graphic" mode: with uppercase letters, numeric digits plus symbols, and 64 graphic characters.
    2. "Business" mode: with lowercase letters, numeric digits plus symbols, uppercase letters, and only 32 graphics characters.
  - Inverted video if the bit #7 of characters is set.
- Peripherals:
  - PIA #1, 6520: Keyboard, vertical retrace interrupt, cassette, IEEE-488
  - PIA #2, 6520: IEEE-488.
  - VIA, 6522: User port, cassette, IEEE-488

This computer has no sound, no joysticks, no cartridges, and no expansion connector. Its memory map is:

		Base address	Size (bytes)	address selection mask
RAM		0x0000	4KB / 8KB[1]	`000n_nnnn_nnnn_nnnn`
video RAM		0x8000	1KB	`1000_xxnn_nnnn_nnnn`
ROM	BASIC	0xC000	8KB	`110n_nnnn_nnnn_nnnn`
	Editor	0xE000	2KB	`1110_0nnn_nnnn_nnnn`
I/O[2]	PIA #1	0xE810	4 registers	`1110_1xxx_xxx1_xxnn`
	PIA #2	0xE820	4 registers	`1110_1xxx_xx1x_xxnn`
	VIA	0xE840	16 registers	`1110_1xxx_x1xx_nnnn`
ROM	Kernel	0xF000	4KB	`1111_nnnn_nnnn_nnnn`

[1]Max 32KB with custom expansions

[2]Notice the incomplete decoding of the peripherals. More than one device could be selected simultaneously.

Replica details

Memory

The ICESTICK board just includes an ICE40HX1K FPGA and a 4MB SPI Flash. The FPGA is rather limited, with only 1280 logic cells and 8KB of synchronous RAM. Just the CPU requires about a 60% of the LCs, and the internal memory is barely enough for a 4KB PET replica because some blocks are used in the video generation logic and keyboard (one BRAM block has 512 bytes):

BRAM usage	Blocks
4KB RAM	8
1KB video	2
2KB character ROM	4
Keyboard matrix	1
Unused	1

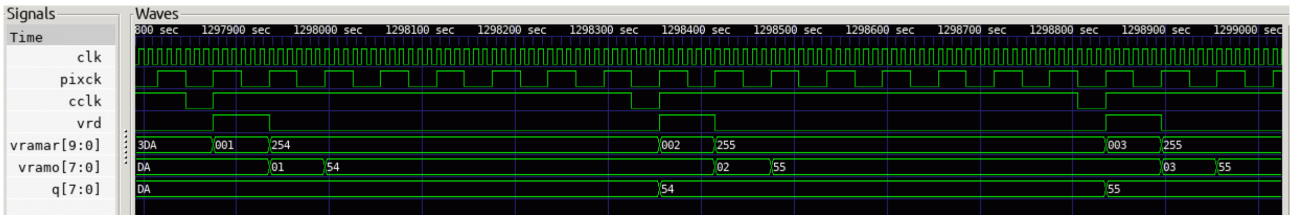
As we can see in this table, 15 out of the available 16 BRAM blocks are used and the replica ROMs aren't included yet. The ROM content is going to be stored in the same flash memory as the FPGA bitstream and read through its SPI bus when needed. There, a random byte read takes no less than 40 clock cycles. Therefore, a 48MHz clock was selected as the main clock, and this signal is routed to the SPI clock when reading the flash. All other clocks are derived from the 48MHz signal, and also this clock is the carrier for the RF modulator logic.

Clocks

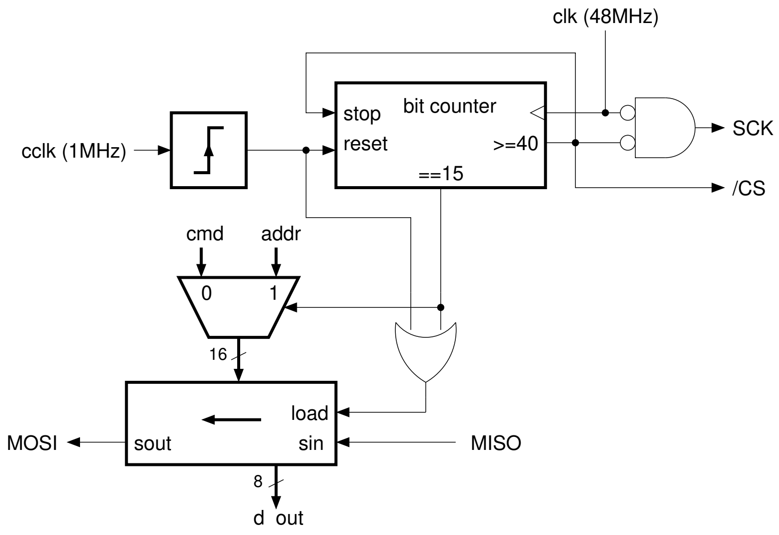
There are several clocks in the replica:

Signal	Frequency	Uses	remarks
clk	48MHz	SPI clock, RF carrier	main clock
pixck	8MHz	Pixel shifting	clk/6
cclk	1MHz	CPU clock	pixclk/8, in phase lock with video reads

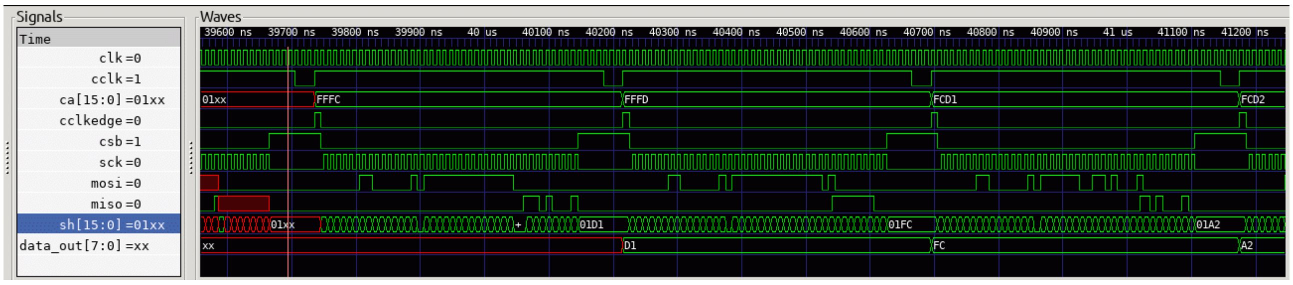
A CPU clock pulse is generated every 8 pixel clock cycles, and this pulse is timed to happen just before a video read. The video data read for the CPU (q[7:0]) is latched at the rising edges of the CPU clock (as it should be for a synchronous memory):



The SPI ROM



The combined ROMs of the PET are stored into the SPI Flash. The address I chose for this image starts at 0x2C000, at the end of the last 64KB block used for the FPGA configuration bitstream (for the HX4K/8K FPGAs, HX1K have smaller bitstreams). The SPI ROM block basically issues a flash read command (0x03) after every rising edge of the CPU clock by means of a 16-bit shift register that gets loaded with this command and the 8 most significant bits of a 24-bit address (0x0302). This register is loaded again after 16 clock pulses with the remaining bits of the address, and keeps shifting in the bits coming out of the flash memory. After 40 cycles the data is available at the lower 8 bits of the shift register, the SPI clock is held low, and the /CS signal deasserted. Not shown in the previous figure is a latching register for the output data that holds the read value during the whole next cclk cycle. This register is needed because the CPU core (a 6502 equivalent by Arlet Ottens) requires a synchronous memory. The timing details are displayed in the following simulation where the reading of the reset vector of the CPU is shown:



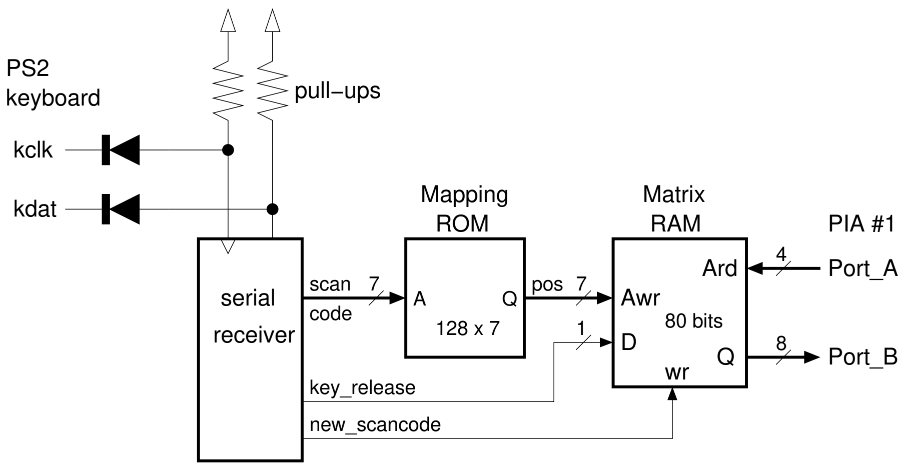
Peripherals

An early version with “decent” PIA an VIA recreations required about 1750 logic cells, way too much. So, these components were replaced by just a minimal recreation of the PIA #1 that allows the connection of a keyboard matrix emulator and the generation of the retrace interrupts, that are needed for keyboard scanning and cursor blinking. The lack of the second PIA means there is no IEEE-488 bus, not a big deal, but the missing VIA will result in no cassette interface nor hardware timers, and this is a more serious limitation. But, in spite of these missing blocks, the BASIC interpreter runs fine.

Keyboard

The PET keyboard is a 10×8 array with the columns driven from a BCD decoder connected to the lower 4 bits of the PIA’s Port A, while the rows are connected to the 8 bits of the Port B. This matrix is simulated by using an 80-bit, dual-port, RAM. Single bits are written with ’0’ when a pressed key scancode is received from the

PS2 keyboard interface, and set to one when the corresponding released key scancode is received. On the read side the 4 lower bits of the PIA's Port A are used as the reading address, and the RAM data is placed on the 8 bits of Port B.



Unfortunately, the symbol to key mapping on the PET is quite different than that of the spanish PS2 keyboard, and some symbols have to be placed on unrelated keys (the hardware is fine, but some keys ought to be relabeled). Otherwise the keyboard works quite well.

Another problem we have to face is the use of 5V logic in the keyboard. Here two possible solutions are:

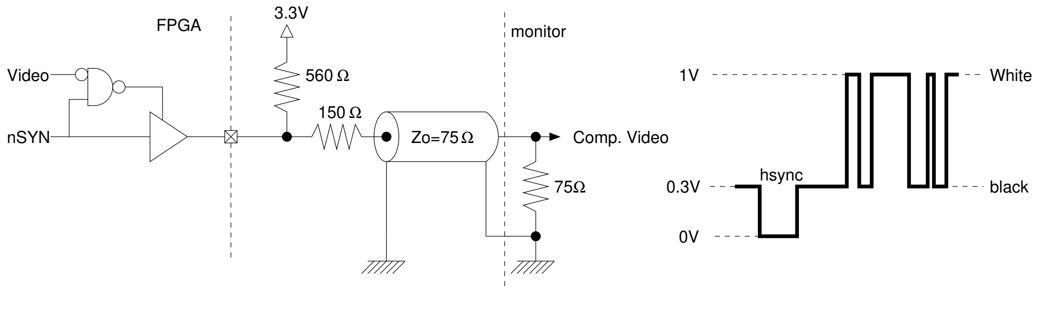
1. Connect the keyboard to a 3.3V supply instead of 5V. Some keyboards have no problem running with lower voltages and their signals can be connected to the FPGA pins directly (my case).

2. Put diodes in series with the two PS2 signals. On the FPGA pins the corresponding pull-ups are enabled, so the default logic level is high. When the keyboard drives a signal low, the corresponding diode will also pull the FPGA pin low, but if the keyboard signal is high, at 5V, the diode is reverse biased, isolating the high voltage from the FPGA pin.

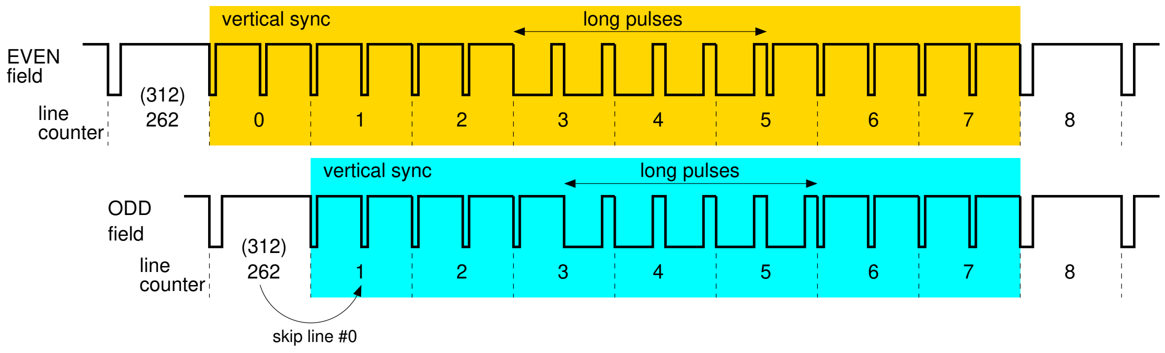
Video generation

The PET has its own video monitor with separate horizontal and vertical sync pulses, but in our recreation we want to use a monitor with a composite video input instead (a TV set). The composite video output has its own complications that we have to face, mainly the generation of a three-level signal and its weird vertical sync waveform, but fortunately there is no color here.

The three-level video signal is generated using just one FPGA pin along with its tristate control in the following way:



The vertical sync waveform is detailed next. Analog TV video is always interlaced, totalling 525 lines for NTSC or 625 lines for PAL. These lines are split into two frame fields: one for the even lines of the image and the next for the odd lines. 8 or 7 lines are reserved for the vertical sync signal as shown in the figure:



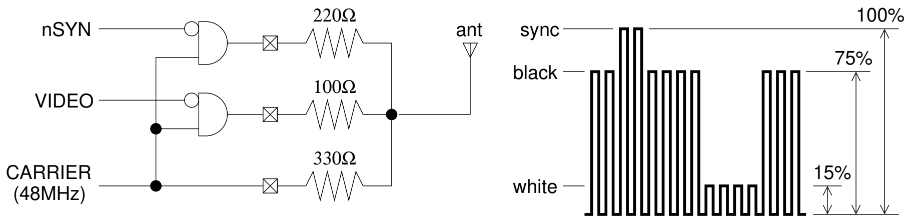
In our recreation the even and odd fields of the image show the same lines. This results in half the vertical resolution, but we only need 200 lines and a full resolution will result in the text being "compressed" vertically at the top of the screen. The displayed lines are placed at the middle of the screen and some top and bottom borders are left around the image.

And with respect to the horizontal resolution, an screen line last $64\mu s$ and its visible time is $48\mu s$. Using the same pixel clock as in the original PET (8 MHz) we can achieve a maximum of 384 pixels per line, but we only need 320 pixels, so, our display time is reduced to $40\mu s$ and some borders are placed at the left and right sides of the screen.

After all the trouble of the sync pulses, the video signal generation is a quite conventional text-mode one, with a 1KB RAM memory for the text and a 2KB ROM for character pixels. The RAM address can come from the CPU or from the vertical and horizontal counters. In this last case the text row (displayed_vertical_line / 8) has to be multiplied by 40 and added to the text column. This multiplication is accomplished using just one adder (40 is 32 plus 8). The output of the video RAM, along with the 3 lower bits of the line counter, is the address input of the character ROM. That ROM is in fact built using BRAMs with a known initial content and write disabled. Well, the bit #7 of the video RAM isn't routed to the ROM address because it is used as a control signal to invert the video output. And also the MSB bit of the character ROM comes from a Graphic/Business input (supposedly controlled by the VIA).

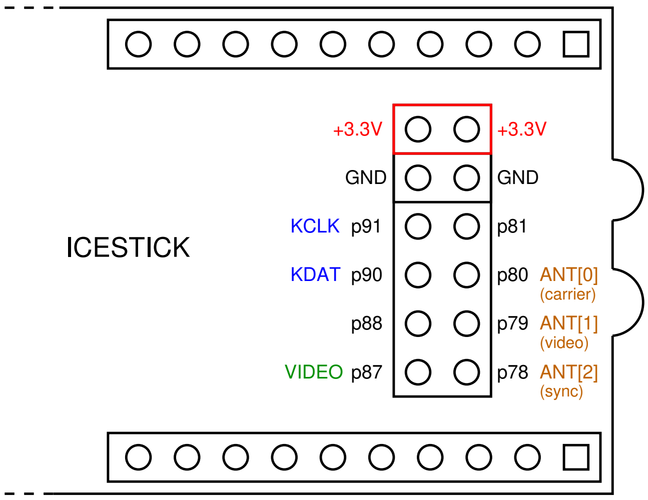
RF generation

Internally, the video and sync are two separate signals. These signals are combined at the output pin to generate the 3-level composite video, but they can also be used along a carrier signal (48MHz) in order to generate a TV signal for an antenna output in the following way:

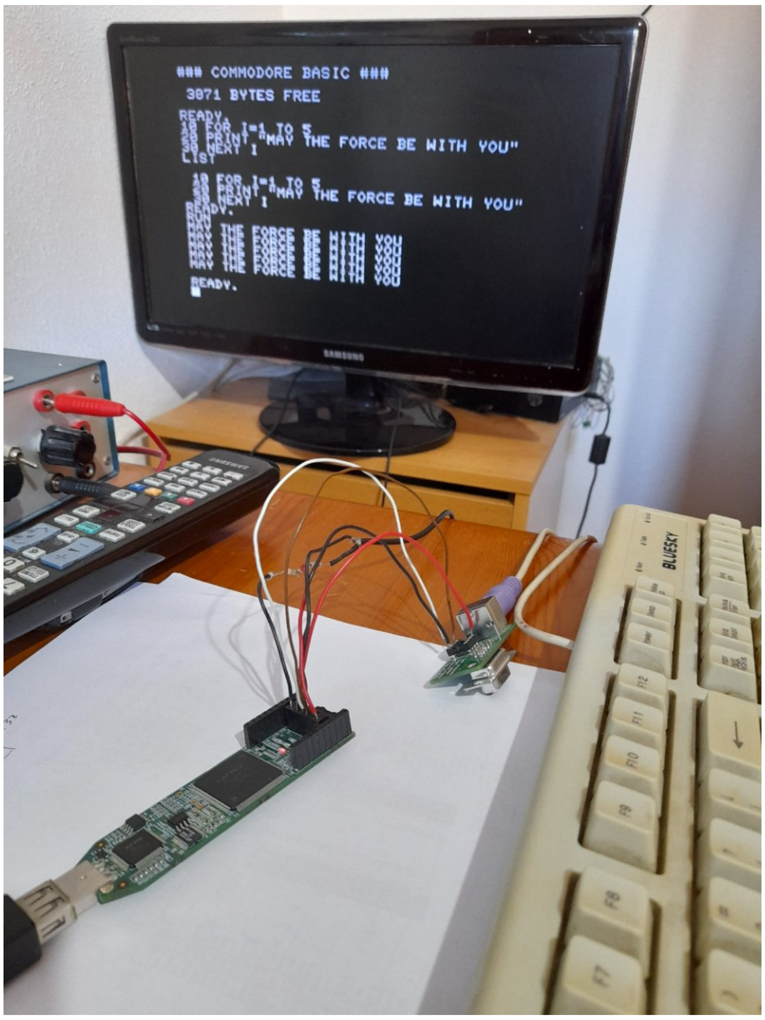


TV video is AM modulated, and this is accomplished by a few gates that translates the video and sync signals into a 4-level analog output thanks to a simple resistor DAC. Three FPGA pins are needed in this case. The resistor ratio must be keep constant because it controls the relative amplitude of the various signal parts, but these resistors can be made higher in order to reduce the RF amplitude at the antenna cable if a direct connection to the TV is desired.

Connections



The signals discussed before are attached to the ICESTICK board pins at the positions shown in the above figure (upper side). A photograph with the board connected to a monitor and a keyboard is included next:



```
### COMMODORE BASIC ###
 3871 BYTES FREE
READY.
10 FOR I=1 TO 5
20 PRINT "MAY THE FORCE BE WITH YOU"
30 NEXT I
LIST

 10 FOR I=1 TO 5
 20 PRINT "MAY THE FORCE BE WITH YOU"
 30 NEXT I
READY.
RUN
MAY THE FORCE BE WITH YOU
MAY THE FORCE BE WITH YOU
MAY THE FORCE BE WITH YOU
MAY THE FORCE BE WITH YOU
MAY THE FORCE BE WITH YOU

READY.
```

A better PET

The presented replica barely fits into the ICE40HX1K FPGA of the ICESTICK board, and only after removing quite a lot of hardware, like the 6522 VIA. Without this chip we have no hardware timers in the PET and there is no way the cassette interface could work at all. But there are also some other boards around using the ICE40HX4K, like Alhambra or ICECREAM, and here we have a lot more logic cells (7680 instead on 1280) and RAM (32 blocks instead of 16). So, here we can do a more realistic replica including decent descriptions of the PIAs and VIA as Verilog modules (code form Thomas Skibo) and we can also replicate a PET with 8KB of RAM instead of only 4KB. And in these replicas we can have a functional cassette interface.

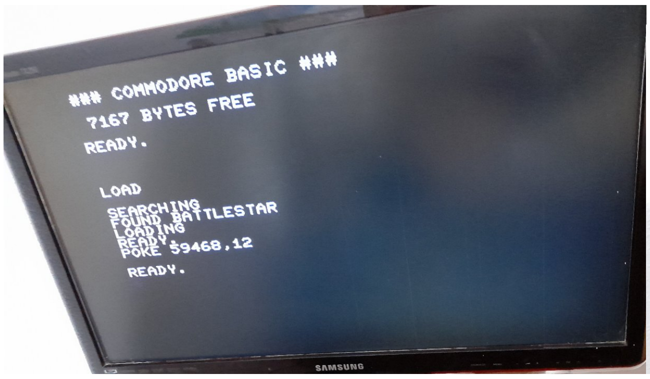
Cassette interface

Well, the replica is now able to generate an audio waveform with the data to save on the VIA pin PB3, and it can also decode an input waveform in PIA #1 pin CA1 or VIA pin CB1 into some data to load. But we have to route these internal signals into some external hardware, and a real cassette player/recorder is out of question (all rubber belts disintegrated long time ago). Some sort of audio input/output is possible, but I wanted to use a communication channel already at hand on these boards, and I chose a serial port as the cassette adapter. With 115200 bps we can transfer 11520 8-bit samples each second and this is enough audio quality for a cassette data tape.

Therefore, the cassette input interface is just a 115200 baud serial receiver with one of the RX buffer bits attached to the cassette inputs. A logic analyzer capture shows the generated wave is composed of square wave cycles of three different lengths: $346\mu s$, $506\mu s$, and $676\mu s$. With a 11520Hz sampling rate these pulses are roughly:

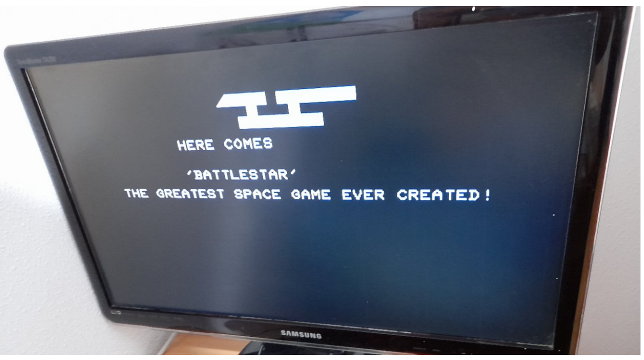
	length (μs)	Samples	actual length (μs)	error %
short	346	`00 00 ff ff`	347.2	+0.3%
medium	506	`00 00 00 ff ff ff`	520.8	+2.8%
long	676	`00 00 00 00 ff ff ff ff`	694.4	+2.7%

The relative time errors are small and probably well below the pulse width variations of a real cassette tape. So, the only remaining problem is the generation of the tape samples, but once this is done we can load some programs into the PET by simple typing “LOAD” and then dumping the samples over the serial port:



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

LOAD
SEARCHING
FOUND BATTLESTAR
LOADING
READY.
POKE 59468,12
READY.
```



```
HERE COMES
'BATTLESTAR'
THE GREATEST SPACE GAME EVER CREATED!
```

For tape output I followed a different approach. Instead of a continuous stream of high or low samples the pulse lengths are measured and transmitted over the serial port. The idea is to follow the same encoding used for the tape files of the VICE emulator, where each byte corresponds to a square-wave pulse of a duration:

$$T_{pulse} = bytevalue \times 8\mu s$$

In this way, not only the data gets compressed with respect to a plain sample stream. Also, if there are no transitions in the cassette output no data is generated at all. And the captured data only needs an header to become a .TAP file that could be used with VICE.

These .TAP files were also the data source for the generation of sample streams for the cassette input, as it was the case of the game shown in the previous photographs.

Audio

Some later PET models included an speaker connected to the CB2 pin of the VIA, so, sound can be generated by means of loading the shift register of the VIA with a data pattern and then recirculating it at a rate controlled by a hardware timer. This capability was also recreated and an audio signal is routed to an FPGA pin.

Some problems

The "reduced" PET recreation always use the "graphic" character set because there is no VIA to control the selection signal, but the "complete" recreation is able to change the character set by means of the following POKEs:

```
POKE 59468,12      for Graphic mode
POKE 59468,14      for Business mode
```

These pokes works as expected, but the PET starts using the "Business" mode after reset and this seems to be wrong. After debugging a little the VIA recreation without finding any problem I tried a different ROM set and it started in "Graphics" mode. Unfortunately, the new ROM set results in a very different keyboard layout, so it was clearly intended for another PET model. So, at the end this is not a problem with the recreation but with the PET firmware itself that defaults to a "Business" character set. After further research I found this is the normal behavior for models with "business" keyboards and Editor ROMs, that is just the kind of matrix I mapped to the PS2 keyboard. I should have selected another Editor ROM and key mapping. Well, I don't want to redo the boring key mapping again, so, at the end we are dealing with a Business PET.

Another possible bug is the cassette motor control signal that starts in an ON state until you execute a "SAVE" or "LOAD" command. In a real cassette player this bug is of little importance because the "Play" key has to be also pressed in order to play or record the tape. It was more annoying when I tried to use this signal as an enable for dumping the cassette data over the serial port after a "SAVE" command. I don't know if this behavior is the same for later ROM revisions, but considering the cassette tapes were completely replaced by floppies during the early eighties it is quite possible to have this bug in all of them.

I must admit the PET ROM set is a sort of chaos with many different versions around for its various parts (Kernel, Editor, and BASIC), and it's easy to chose the wrong ones (BTW, the BASIC version includes the famous "WAIT 6502,n" Easter egg). Anyway, the software repository for the Ubuntu version of VICE, the 8-bit Commodore emulator, says the ROMs are still owned by a Low-Countries company and not included in the emulator. I don't think this is still the case because a recent download of VICE from its web page included all the ROMs, but I wonder why a company wanted to invest into this fossil software (unless that company was also becoming a fossil itself, with no engineers but with lots of lawyers).