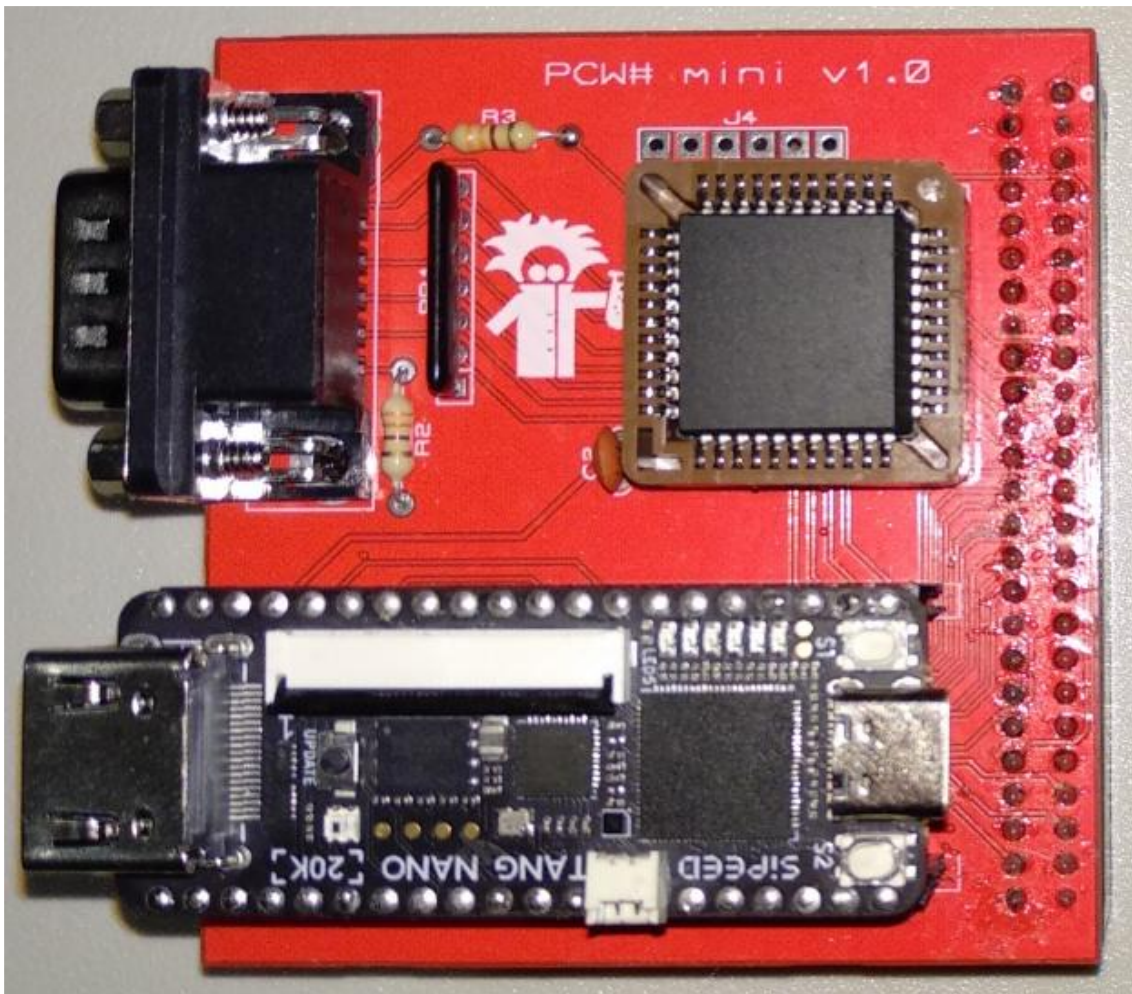


# PCW# mini



User Manual

## Contents

Introduction .....	3
Features.....	6
Definitions .....	6
Hardware .....	7
Software .....	8
ColorIn .....	8
Dk'TurboSound .....	9
Software included .....	11
EdiPal.....	11
ParBios.....	11
Update.....	12
Legal Issues .....	14

# Introduction

PCW# mini is a simplified version of PCW#, a project never released because it was too expensive and complex; also, laziness.

PCW# arises as an experiment; many games, when ported to Amstrad PCW from other platforms, did so with the same graphics if they didn't look too bad. Most Opera games, for example, use the same graphics as their CGA PC versions; the situation in this case is almost identical to the CGA / Hercules relationship of the PC world.

Other companies went further; Level 9 is famous for having released 3" discs that run on ZX Spectrum +3, Amstrad CPC and PCW. On them, the PCW graphics are the same as CPC mode 0 (16 colors).

In any case, the video output in the PCW is classic TTL. We can interpret it as bits and group them in sets of 2 or 4 to obtain graphic modes like those used in the original versions, reinterpreting the signal.



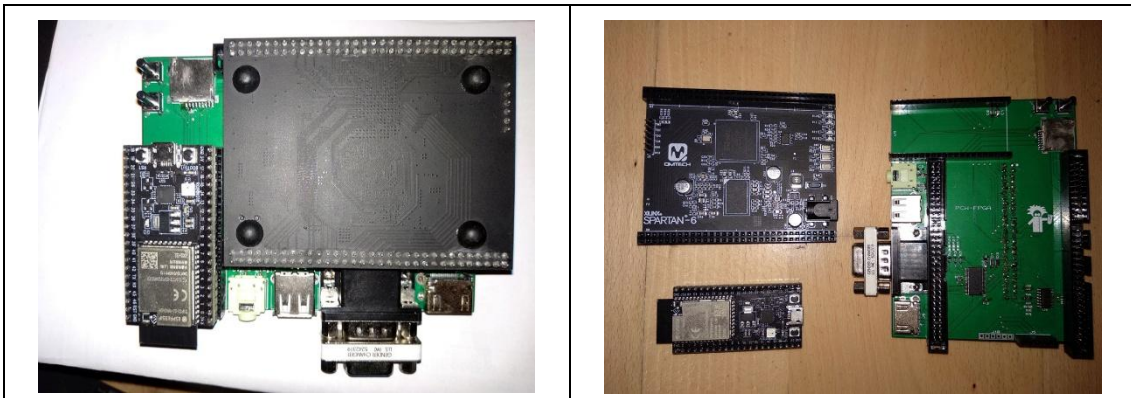
Video reinterpretation examples, 4 and 16 colors.

Being an experiment, an attribute mode was also added, which not only facilitates the porting of software from Spectrum, MSX, Thomson, ... but potentially doubles the speed of the dump to video.

Initially I started testing with a Cyclone II and a simple 6-bit DAC (64 EGA colors). Eventually I migrated to a Spartan 6 with SDRAM (with LVDS support that allows us to encode TMDS, and therefore HDMI) and finally a larger one with DDR. Along the way I added full access to the PCW bus, WiFi, SD, USB, ...



Presentation "in society" of the first prototype. Surely someone recognizes the location.

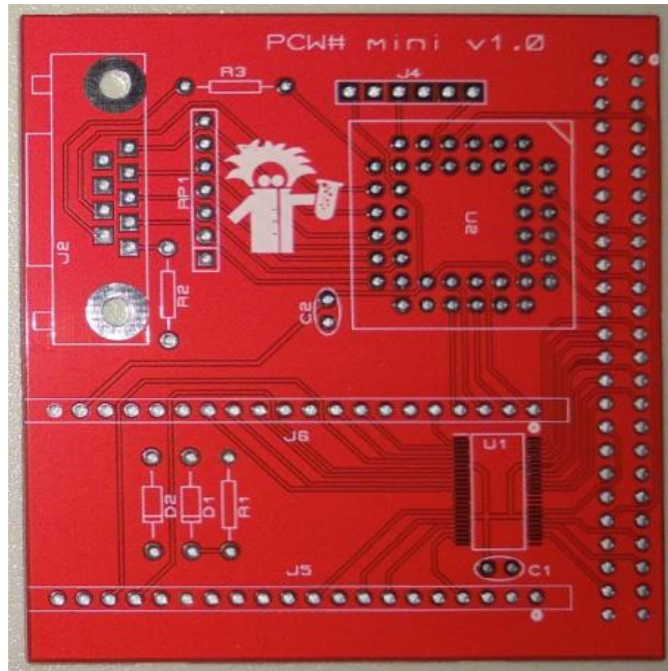


Last prototype, assembled and disassembled.

Given the good reception and the fact that the emulator already had the video modes and interface of the PCW# (including undocumented features), I decided to publish it.

It is not, however, until years later when talking to Enrique G. about the subject I finally decided to release a simplified version using a Tang Nano that I had forgotten in a drawer; the idea was to simplify and create something cheap and easy to assemble, without sacrificing the main thing: offering color modes on a PCW, and then adding as many extras as possible.

And that's how the PCW# mini was born; it is based on a Tang nano 20K, a board that comes with a medium FPGA and comes with an HDMI connector as standard. Although it has few usable exposed legs, the situation is alleviated by using a small CPLD.



Being able to inject audio, a double Dk'tronics card (my stereo variant with DAC) was added, as well as scanlines/CRT effect modes (like the emulator), normal Atari joystick, etc.

Maybe you can't expand memory, intercept boot, provide WiFi, ... But it's also not something you need to enjoy gameplay with improved audio and video, which is its goal.

## Features

PCW# mini offers:

- Plug and play; you don't have to open a PCW or modify anything.
- HDMI output, current digital standard.
- Includes video and audio.
- Complete software control.
- Modes and scanlines can be switched manually.
- Includes the modes: 0-4 of ColorIn.
- Includes default and programmable 24b and 8b (simplified) palettes.
- Includes several scanline modes and CRT filter, like the emulator ones.
- Includes DOUBLE Dk'tronics sound card: two sound chips, two DACs and support for two joysticks. All in stereo.
- Atari standard joystick connector, with support for two buttons (CPC and MSX), as well as for a second joystick with a CPC splitter.
- Upgradeable (including new functionalities) with a simple USB cable.

Limitations:

- The interface expects PAL video, as there were no NTSC PCWs commercially. Support may be added in the future. The idea of forcing it was discarded because it is not something that corresponds to this card.

## Definitions

- Bit: Basic unit of information, with two logical states: "0" and "1". By agreement, they will be written in order from most to least significant if there is more than one.
- Nibble: 4 bits, numbered from 3 to 0.
- Byte: 8 bits, numbered from 7 to 0 (or two nibbles). Any streams will be specified in increasing order of memory or port access.
- PSG: Programmable sound generator, the sound chip used. In the case of the PCW# mini it is the Yamaha YM2149.

# Hardware

The core of the interface consists of a Tang Nano 20K, board with HDMI included (among other things), based on a GW2AR-18 FPGA. Due to the scarcity of usable legs, a CPLD XC9536XL is used as support (address decoding and serialization of the joysticks).

The Joystick connector supports Atari standard, with two buttons (CPC or MSX type), with common on pin 8 or 9 (CPC type selection), and power on 5. Pins 5 and 9 are protected with resistors just in case another type of joysticks are connected that use those pins for other things. At the same time, they can be used for communications, including by current loop (e.g. MIDI).

The interface is powered directly from the PCW, using just under 130 mA. The 5V line is used to power the Tang, and the 3.3V generated by it for the CPLD. There is a protective diode just in case the USB gets connected to a computer with the PCW turned on (or an HDMI device that delivers power).

The S1 and S2 buttons are used to manually switch between graphic mode (with palette switching) and scanlines / filter modes, respectively.

The UPDATE button as well as the 6 normal LEDs are available to the user. The RGB LED indicates status when booting for 2 seconds (red = does not detect PCW, amber = does not detect HDMI, green = all good), and then it is also made available to the user.

The LCD interface and the DAC + amplifier are tuned off due to incompatibility, also the inverter to save power. The SD card (SPI mode), the three additional clocks, access to the Flash and internal features of the FPGA (SDRAM, oscillator, a PLL, possibility of multiboot, etc.) are still available.

Of the two PLLs offered by GW2, one is used to raise the frequency for HDMI. The PCW clock of 32 Mhz is used as the global clock. The design is completely synchronous and latch-free.

At present is not used the possibility of USB OTG with the BL616 included in the Tang. The possibility of using USB PC controllers is being experimented with.

Precautions (other than usual, such as do not connect or disconnect the PCW# mini with the PCW turned on, do not connect it upside down, do not get it wet, etc.):

- It is not recommended to program the CPLD or Tang with the PCW# mini connected to a powered PCW (it is possible, but it is preferable to avoid instabilities).
- Do not connect the Tang upside down on the board (the HDMI goes in the same direction as the joystick).

# Software

From a programmer's point of view, the interface integrates the following devices:

## ColorIn

This device allows the monochrome output of the PCW to be reinterpreted as color modes. It uses two ports, both read and write:

\$80 – Contains the index to the register accessed by port \$81.

\$81 – Contains the register to be accessed.

Registers are grouped by functionality in the top nibble of the index; access, whether read or write, increases the index internally so that data can be easily sent from the Z80 except in specific registers.

No value should be assumed in undefined registers or bits. In writing, an undefined bit must be 0 for future compatibility.

Map of registers:

Group	Index	Function
Video Mode	\$00	Used Video Mode
Palette 24b	\$10-\$1F	24b palette values
Paddle 8b	\$20-\$2F	8b palette values
Border	\$30	24b border value
	\$31	8b border value
	\$32	Border control
Specific PCW# mini	\$FE	RGB LED
	\$FF	Debugging

\$00 – Both read and write: the 6-0 bits specify the video mode used/to be used. At writing, bit 7 indicates if we don't want to change the palette to the default palette (0 = change, 1 = no change). At read, bit 7 is not defined.

\$10 - \$1F – Access to the 24b palette of the corresponding color on the low nibble. 3 accesses are required for each entrance, in order B-G-R.

\$20-\$2F – Access to the simplified palette, one byte per entry. The format is  $R_2R_1R_0G_2G_1G_0B_1B_0$ . Internally it will be converted to 24 bits, both in read and write, truncating/bending the necessary bits ( $R_2R_1R_0G_2G_1G_0B_1B_0 \Leftrightarrow R_2R_1R_0R_2R_1R_0R_2R_1G_2G_1G_0G_2G_1G_0G_2G_1B_1B_0B_1B_0B_1B_0B_1B_0$ ).

\$30 – Same as \$10-\$1F, but for the border.

\$31 – Same as \$20-\$2F, but for the border.

\$32 – Bit 0 controls whether we have the border active or not (1 = active, 0 = not active). The color of the overscan zone will be determined by the color of the edge or entry 0 of the palette, respectively.

\$FE – Works similarly to \$30, but for the Tang's own RGB LED.

\$FF – Debug log; the lower 6 bits correspond to the LEDs of the Tang (1 = off, 0 = on), the 6 on write is the lock of the mode change button (1 = permitted, 0 = blocked) and the 7 on read is the state of the button named "UPDATE". All of this can be useful for debugging on real hardware, or to prevent unwanted interactions.

The video modes are:

Number	Mode (width x height x colors)
0	720x256x2, PCW Original Mode
1	360x256x4
2	180x256x16
3	360x256x16 attributes
4	90x256x256 (unofficial)

All video modes reinterpret the PCW signal. That is, the data is in the order it is displayed, not necessarily the order in memory.

Mode 0 – Same as the original PCW. Each bit represents one pixel, and they are painted from 7 to 0.

Mode 1 – Group the bits in pairs to get 4 colors. 7-6 define the index on the palette for the first, 5-4 for the second, etc. The default palette is the bright CGA BMCW.

Mode 2 – Group the bits in sets of 4 to get 16 colors. 7-4 define the index of the first pixel and 3-0 that of the second. The default palette is the full CGA RGBI range, without changing the dark yellow to brown as the IBM monitor does.

Mode 3 – This is an attribute mode. Two bytes are read, the first is the attributes and the second one the bitmap. The attribute byte contains the indexes for background (bits 7-4) and ink (bit 3-0). The bitmap bits are painted as background (0) or ink (1).

Mode 4 – In this experimental mode each byte is an index on the 8-bit palette, thus achieving 256 colors.

## Dk'TurboSound

This device implements two "Dk'tronics Sound and Joystick" interfaces extended with DAC and stereo, using the IC YM2149 as a base (with respect to the AY-3-8910: double envelope levels, full register bits).

It is recommended to read the corresponding datasheet beforehand for programming the sound chips.

The operation is similar to the original expansion, but it adds PSG selection with \$FE and \$FF values of the latch (TurboSound).

The interface uses 4 ports:

\$A8 – Read the current PSG latch (PCW# mini exclusive).

\$A 9 – Read the value of the current PSG register pointed by the latch.

\$AA – Write the current PSG latch.

\$AB – Write the value of the current PSG register pointed by the latch.

With the latch value \$FF PSG0 is selected, and with \$FE PSG1. Each PSG has a joystick on port A (read) and a DAC of 8b on port B (write).

On boot or reboot, PSG0 will be active in normal mode (ABC stereo + centered DAC). When using the latch to select PSG, TurboSound will be assumed, and the stereo mode will change to PSG0+DAC0 on the left channel and PSG1+DAC1 on the right.

Each joystick has the format:

b7	b6	b5	b4	b3	b2	b1	b0
Fire 2	Fire 1	Top	Down	Right	Left	1	1

This is read via port A (register \$E) of the corresponding PSG. Port B (\$F register) corresponds to their DAC (8b unsigned PCM).

# Software included

## EdiPal

Program to define and set the palette and graphics mode of the PCW# mini. Supports command-line parameters and interactive mode.

Its syntax is as follows:

EdiPal [File] [Mode] [Action]

- [File] is the file with the palette to be loaded.
- [Mode] is the graphics mode (0-4) to set.
- [Action] is the action to be performed: "Y" to return to CP/M and "Z" to restart (same as the keys in interactive mode).

If all three are not specified, it will start into interactive mode. When you start it, the current mode and palette will be read from the hardware; if they are specified on the command line, the latter will have preference.



The interface allows you to define the colors with their index in hexadecimal (0-9, A-F) and values in hexadecimal RGB format, change mode, use the default palette of the mode (with two palettes in the case of mode 0), save the palette (always with the name "pal.bin", it can be renamed in CP/M later), return to CP/M with the chosen palette and mode and simulate a restart with the chosen palette and mode.

## ParBios

Simple BIOS patch that capture BIOS calls to set palette and border color and perform them through PCW# mini. Simply run it and the BIOS will be patched in memory, so as not to alter the EMS file.

# Update

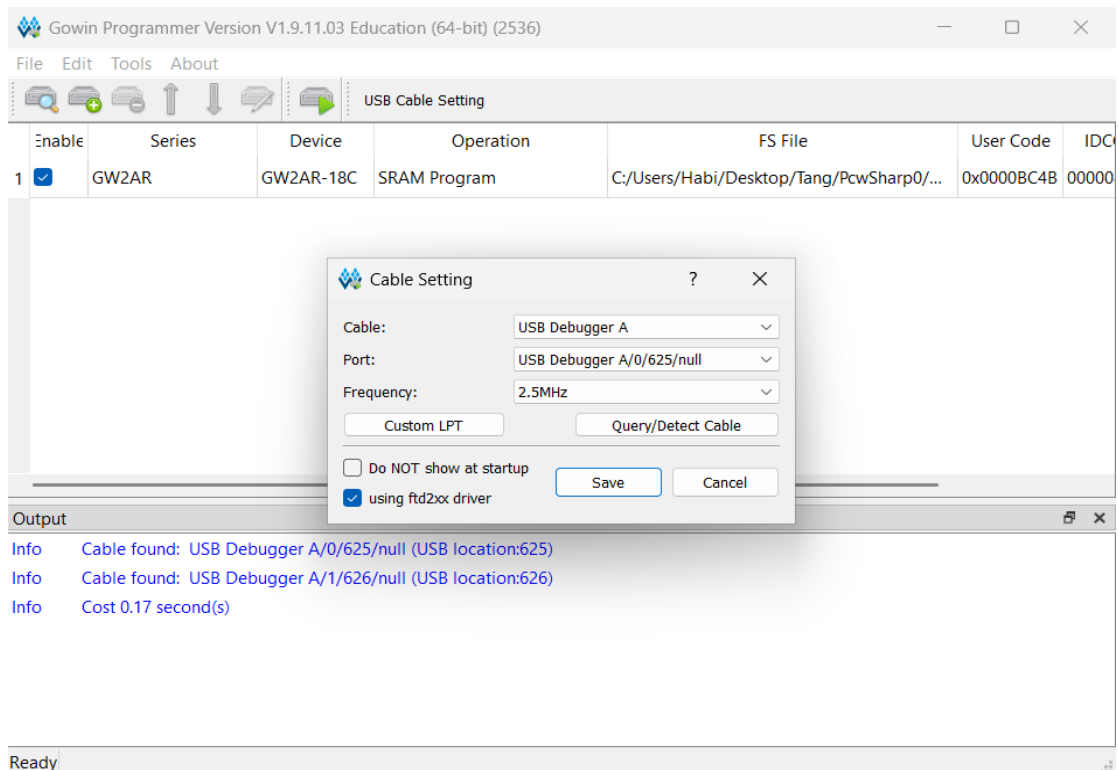
PCW# mini contains three components that are upgradable: the CPLD, the BL616, and the FPGA (the latter two in the Tang itself). Unless there is a radical change, only the FPGA will be updated; it is a simple process and within the reach of any user, which is described below.

To update you need:

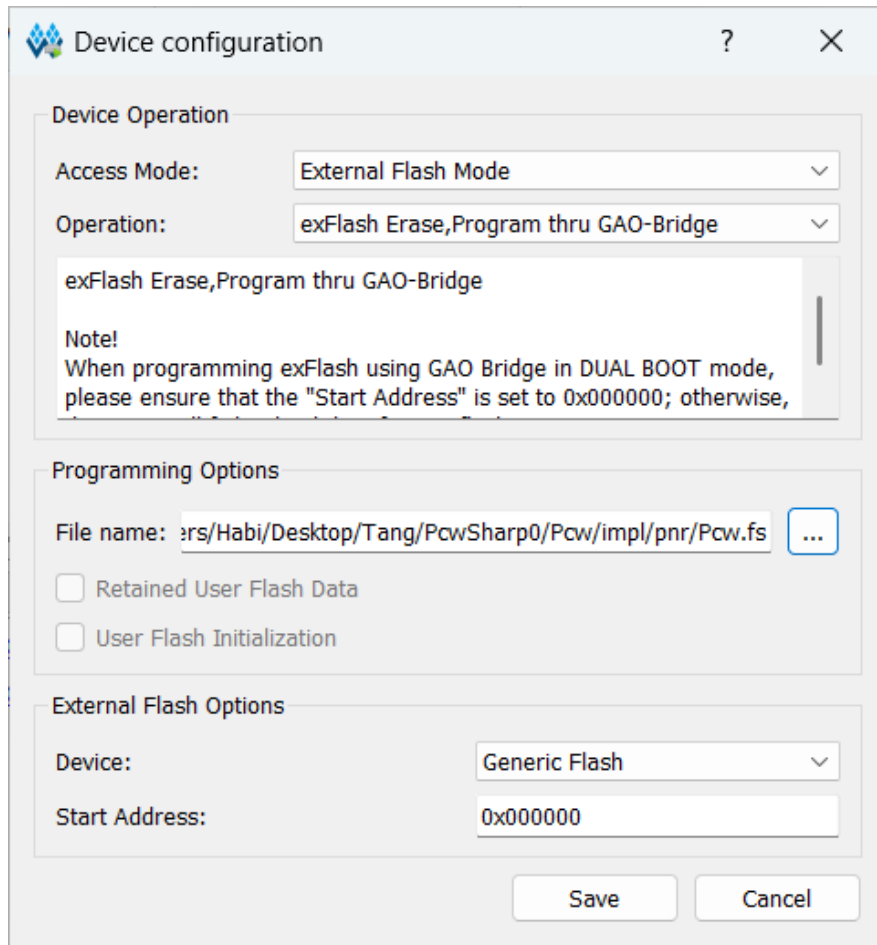
- 1) The Tang board, preferably disconnected from the PCW# mini.
- 2) A USB cable (e.g. the one included with the Tang).
- 3) The "[GOWIN Programmer](#)" program installed on a PC, for its corresponding operating system.

The steps to update it are as follows:

- 1) Download the file corresponding to the version we want to install.
- 2) Connect the Tang with the USB cable to the PC.
- 3) Open the program; it should detect the FPGA:



- 4) Choose the "External flash" recording option in the program as the following capture, select the file and select save:



5) Done. You can now disconnect the USB cable, close the program, etc.

## Legal Issues

All trademarks or rights of any kind, registered names or logos, are the property of their respective owners.