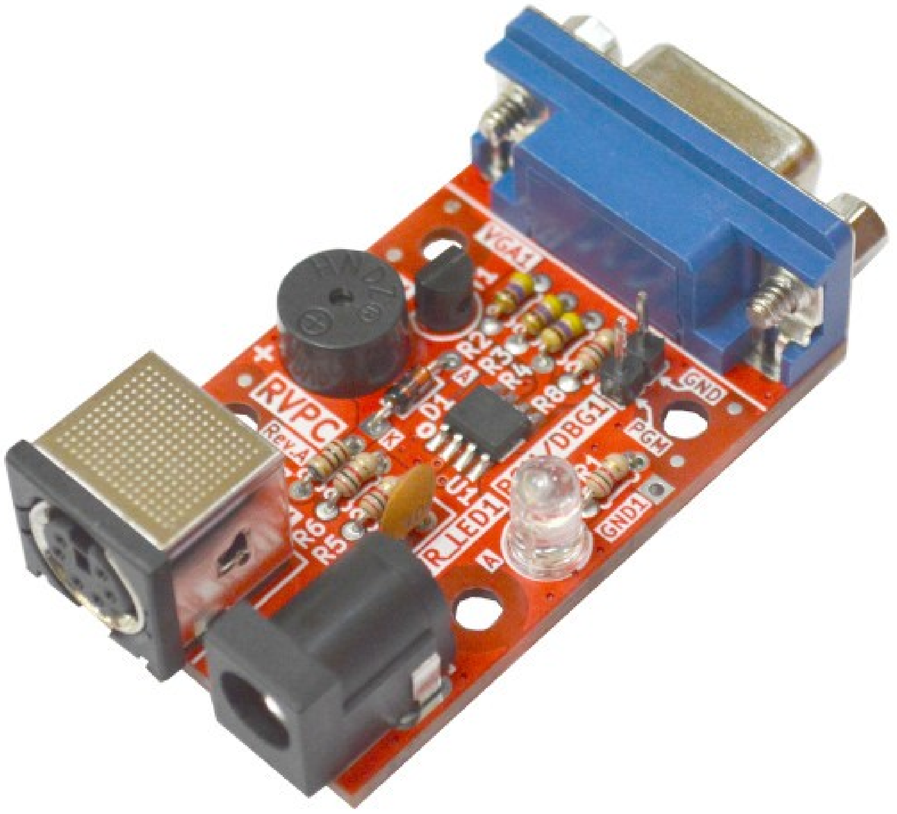


RVPC

User Manual

olimex.com

Rev.1.0   June 2024

Table of Contents

What is RVPC?

RVPC is an attempt to produce very low cost EURO 1.00 educational computer with RISC-V processor which to have everything one complete computer have: Keyboard input, VGA display output and Audio output.

The idea of RVPC evolved on TuxCon 2024 as a Lighting talk https://youtu.be/YlYE9a7zsqY.

The goal set was:

1. Easy to solder DIY kit
2. Complete all in one RISC-V computer with bare minimum Woz like monitor which will allow you to learn the RISC-V instructions by poking, peeking and disassembling the memory
3. Price of EUR 1.00!

Here is the result:

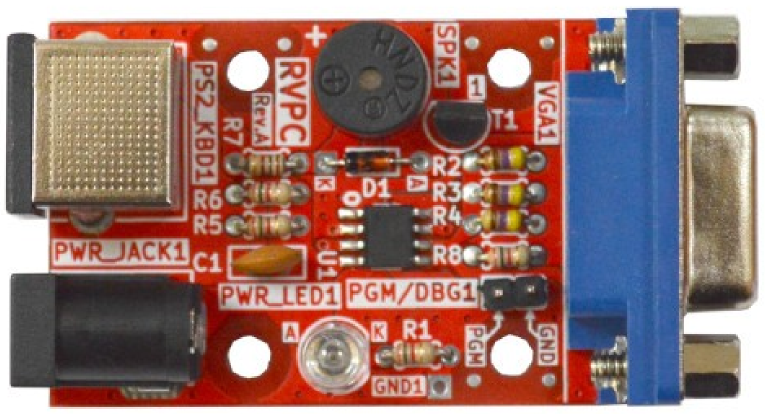
CH32V003 in SO8 package – for easy soldering was chosen. It has just 6 GPIOs

- PS2 takes two GPIOs

- VGA takes three GPIOs – Vsync, Hsync and RGB

- Audio buzzer is connected to the last GPIO

All done in beginner friendly PTH components

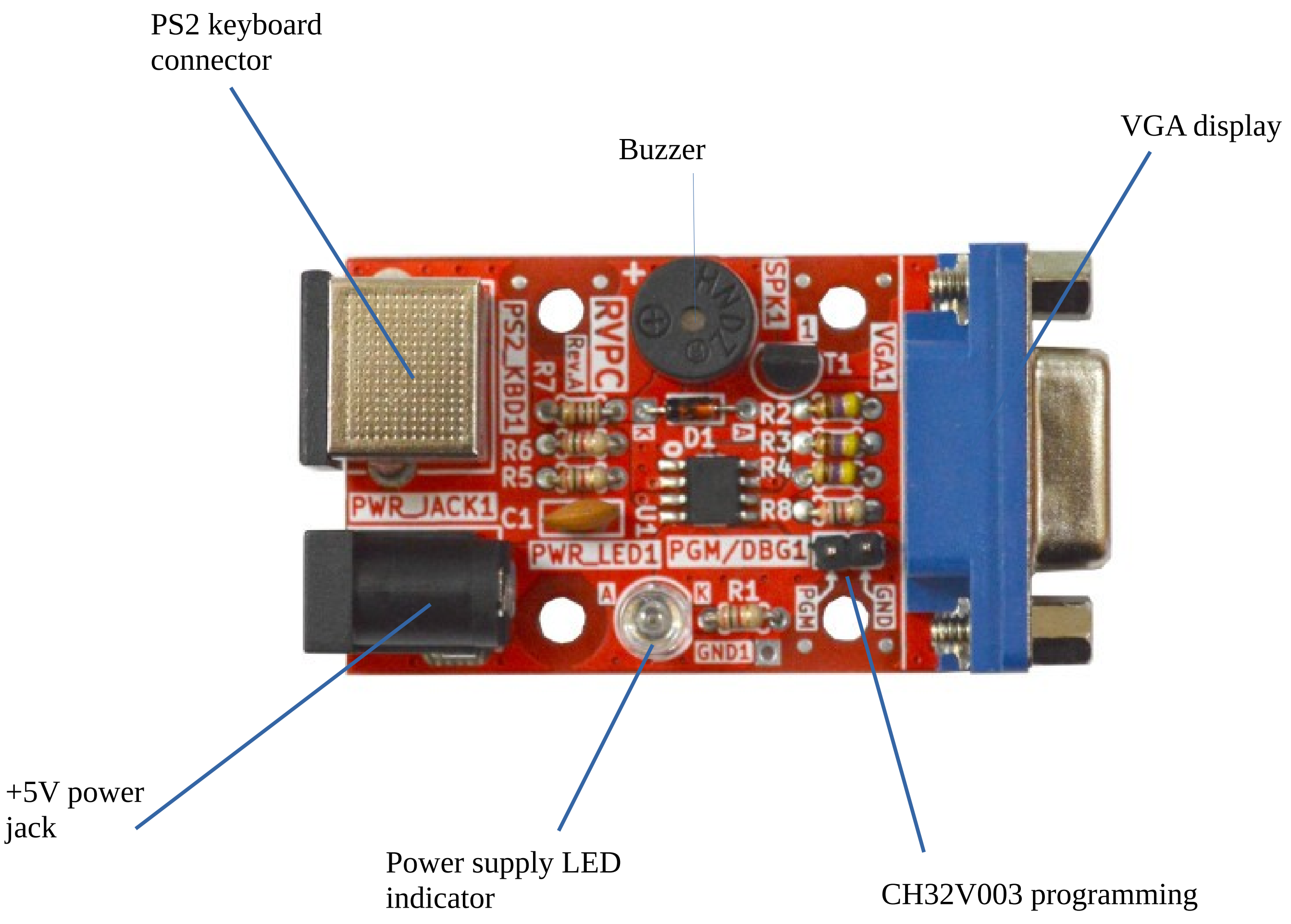


Order codes for RVPC and accessories:

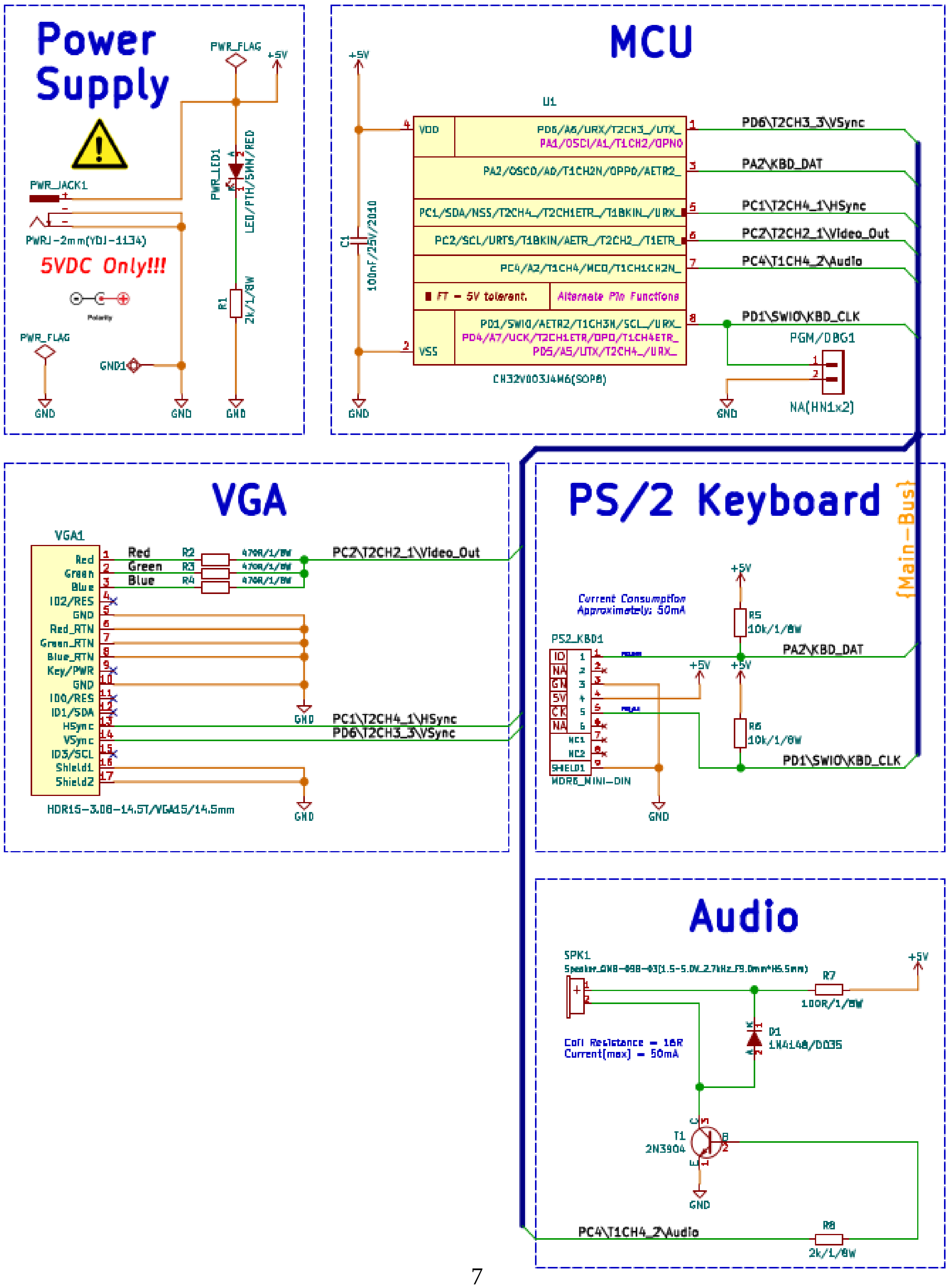
RVPC	Do It Yourself soldering kit
SY0605E	5V power supply adapter
PS2-Keyboard	PS2 keyboard
ESP32-S2-DevKitLiPo-USB	ESP32-S2 development board which can be used as CH32V003 programmer

HARDWARE

RVPC layout:


PS2 keyboard connector
Buzzer
VGA display
+5V power jack
Power supply LED indicator
CH32V003 programming

RVPC schematic:



SOFTWARE:

Here below is our setup under Linux:

Install - packages

$ apt-get install build-essential libnewlib-dev gcc-riscv64-unknown-elf libusb-1.0-0-dev libudev-dev gdb-multiarch

Install - Visual Studio Code

Described here: https://code.visualstudio.com/docs/setup/linux

Install - Platform IO

Described here: https://platformio.org/install/ide?install=vscode

Install - CH32V-Platform

https://github.com/Community-PIO-CH32V/ch32-pio-projects?tab=readme-ov-file#installing-the-ch32v-platform

by default the platformio generates only .elf file, to build firmware.bin and firmware.elf select

> PlatformIO > PROJECT TASKS > Default > Advanced > Verbose build

Sample Beeper project is in RVPC repository.

Prepare the CH32V003 programmer

ESP32-S2-DevKitLiPo-USB can be used as programmer.

The firmware is here you can build from sources or you can download the ready built binaries from here and use this sequence to prepare the programmer:

1. hold the Boot button and connect the USB cable, the yellow LED will stay ON

check with

```
$ls /dev/ttyA*
```

which is the ttyACM it’s usually 0 or 1

execute this command:

```
$ python3 ./rvpc/esptool/esptool.py -p /dev/ttyACM0 -b 460800 --before=no_reset --after=no_reset
write_flash --flash_mode dio --flash_freq 80m --flash_size 4MB 0x1000 ./rvpc/esp32s2/bootloader.bin
0x10000 ./rvpc/esp32s2/usb_sandbox.bin 0x8000 ./rvpc/esp32s2/partition-table.bin
```

check if the programmer is already OK with

```
$ dmesg
```

you have to see this message:

```
hid-generic 0003:303A:4004.0015: input,hidraw5: USB HID v1.11 Gamepad [CNLohr ESP32-S2
CH32V003Programmer] on usb-0000:00:14.0-2/input0
```

which means the ESP32-S2-DevKitLipo-USB now act as programmer and can be used with the demo project above from PlatformIO, but first you have to enable it with:

```
$ sudo cp ./rvpc/tools/ch32v003fun/minichlink/99-minichlink.rules /etc/udev/rules.d/
$ sudo udevadm control --reload-rules && sudo udevadm trigger
```

Now you can use GPIO6 and GND to connect to RVPC programming connector PGM-GND

Now CH32V003 flashing will work directly from PlatformIO but if you want to use command line this is the command:

```
./rvpc/tools/ch32v003fun/minichlink/minichlink -w ./firmware.bin 0x08000000
```

Using Raspberry Pi Pico as a programmer

Raspberry Pi Pico can be used as a programmer.

The programmer firmware is located here: https://github.com/aappleby/picorvd. Make sure you install the dependencies, especially GDB as you'll need it for uploading the RVPC firmware later.

You can either build it yourself, or use a precompiled binary from their CI. The prebuilt binaries can be found in the **Actions** tab on the repository, selecting the most recent commit and scrolling to the bottom of the page where the **Artifacts** section is. There should be a single artifact named **PicoRVD.uf2**, which is the firmware build. If the downloaded file is a ZIP, make sure to unzip the firmware and discard the ZIP file.

Get your RPi Pico, hold down the BOOTSEL button and plug the USB cable into your computer. A new USB mass storage device should appear, copy the PicoRVD firmware file into it and wait for a moment. You don't need to safely eject/remove it, since it will disconnect itself as soon as it receives the firmware file. It will then immediately boot the PicoRVD firmware and the Pico will be ready for flashing your RVPC.

RVPC firmware uploading is done by the Remote debugging feature of GDB. In order to automate it, you can use the following shell script:

```
#!/bin/bash

set -e

if [ -z "${1}" ]
then
   echo 'usage: '"${0}" path/to/firmware.elf >&2
   exit 1
fi

gdb-multiarch -ex 'target extended-remote /dev/ttyACM0' -ex 'load' -
ex 'detach' -ex 'quit' "${1}"
```

To upload a new RVPC firmware, you need the .elf file, not the .bin file as GDB does not have enough information how to upload it. Before uploading a new firmware, restart your RVPC by power cycling it, then power-cycle your Pico and in case it appears as mass storage device, upload the PicoRVD firmware again. Then use the shell script, adjusting the /dev/ttyACM0 port according to the actual serial port name (you can find it in dmesg). After successfully uploading a new RVPC firmware, you need to power-cycle the RVPC, in order to start the program.

Create project:

If you create new project to enable the ESP32-S2 programmer you should edit platformio.ini and add this line

```
upload_protocol = minichlink
```

It’s already added to the demo project.

Revision History

Revision 1.0 June 2024