USER MANUAL

RVT101HVC CM4 series

Raspberry Pi CM4 setup on Raspberry Pi OS
Rev. 1.0
2026-09-17

Riverdi Sp. z o.o.

Nowy Swiat 36
80-299 Gdansk, Poland
VAT ID: PL5842767153
Registration number (KRS): 0000923967

+48 587 703 116

Office hours: 9:00 to 17:30 CET (Mon-Fri)

contact@riverdi.com

riverdi.com

ITEM	CONTENTS	UNIT
Display module	RVT101HVC series	/
Compute module	Raspberry Pi CM4	/
Operating system	Raspberry Pi OS (64-bit)	/
Resolution	1280 × 800 at 60 Hz	/
Power supply (VIN)	7.5 – 55 (12 recommended)	V
Power control	PWR_EN input, PoE module header	/
Video	Internal LCD, external HDMI	/
Communication	Ethernet, USB, 2× CAN, RS232, RS485/RS422	/
Debug	UART console	/
Storage / expansion	M.2 NVMe, M.2 modem with SIM, microSD	/
Camera / audio	2× camera, speaker output	/

1. REVISION RECORD

REV NO.	REV DATE	CONTENTS
1.0	2026-09-24	Initial release

2. CONTENTS

1. REVISION RECORD 3
2. CONTENTS 4
3. INTRODUCTION 5
3.1 Purpose and Scope 5
3.2 Tested Operating Systems 5
3.3 Other Operating Systems 5
3.4 Required Hardware and Software 5
4. HARDWARE INTERFACES 6
4.1 Module Interfaces 6
4.2 Display, Touch and Brightness 7
4.3 GPIO Assignment 7
5. INSTALLATION 8
5.1 Overview 8
5.2 Step 1 – Install rpiboot (CM4 with eMMC) 8
5.3 Step 2 – USB Boot Mode (CM4 with eMMC) 9
5.4 Step 3 – Write Raspberry Pi OS 9
5.5 Step 4 – Start the Module and Update the System 10
5.6 Step 5 – Configure the Display and the Touch Panel 10
5.7 Step 6 – Install pigpio 11
5.8 Step 7 – Install the Brightness Control Application 12
5.9 Step 8 – Reboot and Verify 13
5.10 Series Deployment 13
6. VERIFICATION 13
6.1 System Information 13
6.2 Display 14
6.3 LCD Control Signals 14
6.4 Touch 14
6.5 Brightness 14
6.6 I²C Devices (Optional) 14
7. APPLICATION BRIGHTNESS CONTROL 15

3. INTRODUCTION

3.1 Purpose and Scope

This manual describes how to configure a clean Raspberry Pi OS image so that the RVT101HVC series display module operates with a Raspberry Pi Compute Module 4 (CM4). It covers the LCD, the touch panel and the backlight brightness control.

All settings use standard Raspberry Pi OS components. No kernel modules or drivers are installed, so the procedure can be repeated on new OS releases and automated for series deployment.

3.2 Tested Operating Systems

COMPUTE MODULE	OPERATING SYSTEM	STATUS
Raspberry Pi CM4	Raspberry Pi OS (64-bit) Trixie	Tested
Raspberry Pi CM4	Raspberry Pi OS (64-bit) Bookworm	Tested

The installation procedure in Section 5 is identical for both releases.

3.3 Other Operating Systems

The display module does not require Raspberry Pi OS. The LCD is connected to a standard HDMI output, the touch panel is a standard USB HID multi-touch device, and the control signals are plain GPIOs. Any operating system that runs on the CM4 can operate the module when it provides the functions below.

FUNCTION	REQUIREMENT	IMPLEMENTATION
LCD enable	GPIO22 and GPIO23 driven high	`gpio=` lines in `config.txt` (Step 5); read by the CM4 boot firmware for every operating system
LCD image	HDMI0 at 1280 × 800, 60 Hz	`hdmi_*` lines in `config.txt` (Step 5), or `video=HDMI-A-1:1280x800@60D` on the kernel command line
Touch	USB 2.0 host, HID multi-touch driver	Kernel with DWC2 USB host and `hid-multitouch`
Brightness	16-bit write on GPIO13, GPIO20, GPIO21	Any GPIO access method (Section 7)

3.4 Required Hardware and Software

- RVT101HVC series display module.
- Raspberry Pi CM4, eMMC or Lite variant.
- DC power supply, 7.5 – 55 V (12 V, 2 A recommended).

- Host PC (Windows or Linux) with Raspberry Pi Imager; for a CM4 with eMMC also the Raspberry Pi `rpiboot` utility (Step 1).
- CM4 with eMMC: USB cable to connect the USB boot connector P1 of the module to the host PC.
- CM4 Lite: microSD card (16 GB or larger) and a card reader for the host PC.
- Network connection (Ethernet or Wi-Fi) for SSH access and package installation.

4. HARDWARE INTERFACES

4.1 Module Interfaces

INTERFACE	CM4 RESOURCE / FUNCTION
DC power input	7.5 – 55 V DC
Power enable input PWR_EN (J18)	Enables the on-board power supplies. Open: module on (pulled up on the module). Connected to GND: module off
PoE module header	Connection for an optional PoE power module
Internal LCD	HDMI0
External HDMI	HDMI1
Ethernet	CM4 Gigabit Ethernet
USB ports (2×)	USB 2.0 through the on-board USB hub
M.2 modem (B-key) with SIM	USB 2.0 through the on-board USB hub
M.2 NVMe (M-key)	PCIe
microSD (CM4 Lite)	SDIO
CAN (2×)	SPI1, interrupts on GPIO10 and GPIO25
RS232	UART3
RS485/RS422	UART4
Debug console	UART0
Camera (2×)	CAM0, CAM1
Speaker output	HDMI0 audio through the display bridge
RTC	I²C1
Fan connector	Fan controller on I²C1
IO expander headers (2×)	IO expander on I²C1
Expansion header	SPI1, I²C1
Buttons BOOT (SW1), EN (SW2), PWR (SW3)	CM4 USB boot mode, CM4 global enable, CM4 reset

4.2 Display, Touch and Brightness

FUNCTION	CM4 INTERFACE	CONFIGURATION
LCD image	HDMI0 (Linux connector HDMI-A-1)	Fixed mode 1280 × 800, 60 Hz – Step 5
LCD reset	GPIO22	Output, high – Step 5
LCD standby	GPIO23	Output, high – Step 5
Touch	USB 2.0 through the on-board USB hub	USB host (DWC2) – Step 5; both jumpers fitted on P2
Brightness	GPIO13, GPIO20, GPIO21	Brightness control application – Steps 6 and 7

4.3 GPIO Assignment

GPIO	FUNCTION
GPIO0, GPIO1	I²C for camera connector CAM0
GPIO2, GPIO3	I²C1 – on-board devices
GPIO4 – GPIO7	UART3 – RS232
GPIO8, GPIO9, GPIO11	UART4 – RS485/RS422 (GPIO11: driver enable)
GPIO10, GPIO25	CAN controller interrupts
GPIO12	Not used
GPIO13, GPIO20, GPIO21	Backlight brightness
GPIO14, GPIO15	UART0 – debug console
GPIO16 – GPIO19	SPI1 – CAN controllers and expansion header
GPIO22, GPIO23	LCD reset and standby
GPIO24	Audio mute
GPIO26, GPIO27	Touch controller interrupt and reset

Keep GPIO13 and GPIO20 – GPIO23 unchanged in your application. SPI0, PWM on GPIO18/GPIO19 and I²S are not available on this module.

5. INSTALLATION

5.1 Overview

The installation path depends on the CM4 variant. The variant is shown in the CM4 part number: the last three digits give the eMMC size, and `000` identifies a CM4 Lite (for example, CM4104032 has 32 GB eMMC, CM4104000 is a CM4 Lite).

CM4 VARIANT	OPERATING SYSTEM STORAGE	STEPS
CM4 with eMMC	On-board eMMC, written from the host PC through USB boot mode	Steps 1, 2, 3A, 4 – 8
CM4 Lite	microSD card, written on the host PC with a card reader	Steps 3B, 4 – 8

Overview of the steps:

1. Install rpiboot on the host PC (CM4 with eMMC).
2. Connect the module in USB boot mode (CM4 with eMMC).
3. Write Raspberry Pi OS: to the eMMC (3A) or to a microSD card (3B).
4. Start the module and update the system.
5. Configure the display and the touch panel.
6. Install pigpio.
7. Install the brightness control application.
8. Reboot and verify.

5.2 Step 1 – Install rpiboot (CM4 with eMMC)

Install rpiboot using the method for the operating system of the host PC:

HOST PC	INSTALLATION
Windows	Download and run the installer `rpiboot_setup.exe` from the Releases page of https://github.com/raspberrypi/usbboot. After installation, rpiboot is available in the Start menu.
Raspberry Pi	Install the prebuilt package: `sudo apt install -y rpiboot`
Other Linux distributions	Build rpiboot from source as described below.

Building rpiboot from source (Linux):

1. Install the build dependencies:

```
sudo apt install -y git libusb-1.0-0-dev pkg-config build-essential
```

2. Download and build rpiboot:

```
git clone --depth=1 https://github.com/raspberrypi/usbboot
cd usbboot
make
```

5.3 Step 2 – USB Boot Mode (CM4 with eMMC)

The following parts of the module are used in this step:

LABEL	PART	USE IN THIS STEP
P1	USB boot connector	Connection to the host PC
P2	2 × 2 pin header with two jumpers	Jumpers removed during USB boot mode
SW1	Push-button marked BOOT	Held while the power supply is connected

1. Disconnect the power supply from the module.
2. Remove both jumpers from header P2 and keep them for Step 3A.
3. Connect the USB boot connector P1 to a USB port of the host PC.
4. Start rpiboot on the host PC and leave it running. rpiboot waits for the module:

```
Windows:                          start rpiboot from the Start menu
Raspberry Pi (package):           sudo rpiboot
Linux (source build, in usbboot): sudo ./rpiboot
```

5. Press and hold the BOOT button SW1. Do not press SW2 (EN) or SW3 (PWR).
6. While holding SW1, connect the power supply.
7. Keep SW1 pressed until rpiboot prints `Sending bootcode.bin`, then release SW1.
8. Wait until rpiboot has finished. The eMMC of the CM4 appears on the host PC as a new USB drive within approximately 10 seconds.
9. On Windows, close any prompt that offers to format the drive without formatting.

5.4 Step 3 – Write Raspberry Pi OS

3A – CM4 with eMMC:

1. Start Raspberry Pi Imager on the host PC.
2. Under **Device**, select **Raspberry Pi 4** (this selection includes the Compute Module 4).
3. Under **Operating System**, select **Raspberry Pi OS (64-bit)** with desktop.
4. Under **Storage**, select the USB drive of the CM4 eMMC from Step 2.
5. Open the OS customisation settings and set the hostname, user name and password, network (Wi-Fi if used), time zone and keyboard layout.
6. In the OS customisation settings, enable SSH.
7. Start writing and confirm the prompt. Wait until writing and verification are complete.
8. Disconnect the power supply from the module.
9. Disconnect the USB cable from P1.
10. Refit both jumpers on header P2.

3B – CM4 Lite:

1. Insert the microSD card into the card reader of the host PC.
2. Start Raspberry Pi Imager on the host PC.
3. Under **Device**, select **Raspberry Pi 4** (this selection includes the Compute Module 4).
4. Under **Operating System**, select **Raspberry Pi OS (64-bit)** with desktop.
5. Under **Storage**, select the microSD card.
6. Open the OS customisation settings and set the hostname, user name and password, network (Wi-Fi if used), time zone and keyboard layout.
7. In the OS customisation settings, enable SSH.
8. Start writing and confirm the prompt. Wait until writing and verification are complete.
9. Remove the microSD card from the card reader.
10. Check that the power supply of the module is disconnected and that both jumpers are fitted on header P2.
11. Insert the microSD card into the microSD slot of the module.

5.5 Step 4 – Start the Module and Update the System

1. Connect the network cable if Ethernet is used.
2. Connect the power supply and wait approximately one minute for the first boot to complete.
3. Log in from the host PC via SSH, using the hostname and user name set in Step 3:

```
ssh <user>@<hostname>.local
```

4. Update the package lists and installed packages:

```
sudo apt update
sudo apt full-upgrade -y
```

5. Reboot the module and log in again via SSH:

```
sudo reboot
```

5.6 Step 5 – Configure the Display and the Touch Panel

1. Create backup copies of the boot configuration files:

```
sudo cp /boot/firmware/config.txt /boot/firmware/config.txt.bak
sudo cp /boot/firmware/cmdline.txt /boot/firmware/cmdline.txt.bak
```

2. Disable the default USB controller setting:

```
sudo sed -i 's/^otg_mode=1/#otg_mode=1/' /boot/firmware/config.txt
```

3. Remove any HDMI mode parameter from the kernel command line:

```
sudo sed -i -E 's/ ?video=HDMI-A-1:[^ ]*//' /boot/firmware/cmdline.txt
```

4. Remove a Riverdi settings block from a previous installation, if present:

```
sudo sed -i '/^# >>> riverdi-rvt101/,/^# <<< riverdi-rvt101/d' \
  /boot/firmware/config.txt
```

5. Append the Riverdi settings block to the end of `config.txt`:

```
sudo tee -a /boot/firmware/config.txt > /dev/null << 'EOF'
# >>> riverdi-rvt101
[all]
gpio=22=op,dh
gpio=23=op,dh
hdmi_group=2
hdmi_mode=87
hdmi_cvt=1280 800 60 6 0 0 0
hdmi_drive=2
dtoverlay=dwc2,dr_mode=host
# <<< riverdi-rvt101
EOF
```

6. Check the result. The output must show `#otg_mode=1` and the Riverdi settings block:

```
grep -n 'otg_mode' /boot/firmware/config.txt
tail -n 11 /boot/firmware/config.txt
```

SETTING	PURPOSE
`gpio=22=op,dh`	Sets GPIO22 (LCD reset) as output, high
`gpio=23=op,dh`	Sets GPIO23 (LCD standby) as output, high
`hdmi_group=2`, `hdmi_mode=87`, `hdmi_cvt=1280 800 60 6 0 0 0`	Sets HDMI0 to 1280 × 800 at 60 Hz
`hdmi_drive=2`	Selects HDMI signalling
`dtoverlay=dwc2,dr_mode=host`	Enables the USB 2.0 host used by the on-board hub and the touch panel

5.7 Step 6 – Install pigpio

1. Install the build tools:

```
sudo apt install -y build-essential unzip wget python3-setuptools
```

2. Download, build and install pigpio:

```
cd ~
wget -O pigpio.zip https://github.com/joan2937/pigpio/archive/refs/heads/master.zip
unzip pigpio.zip
cd pigpio-master
make
sudo make install
```

3. Create the pigpio service. The option `-l` limits access to local applications:

```
sudo tee /etc/systemd/system/pigpiod.service > /dev/null << 'EOF'
[Unit]
Description=pigpio daemon

[Service]
Type=forking
ExecStart=/usr/local/bin/pigpiod -l -p 8888

[Install]
WantedBy=multi-user.target
EOF
```

4. Enable and start the service:

```
sudo systemctl daemon-reload
sudo systemctl enable --now pigpiod
```

5. Check the service state. The output must be `active`:

```
systemctl is-active pigpiod
```

5.8 Step 7 – Install the Brightness Control Application

The brightness control application opens a window with a slider and sets the backlight brightness through the pigpio service.

1. Install PyQt5:

```
sudo apt install -y python3-pyqt5
```

2. Create the application file:

```
sudo tee /usr/local/bin/riverdi-brightness.py > /dev/null << 'EOF'
#!/usr/bin/env python3
# RVT101HVCNWC00-B backlight brightness control
import sys
import pigpio
from PyQt5.QtCore import Qt
from PyQt5.QtWidgets import QApplication, QLabel, QSlider, QVBoxLayout, QWidget

CS, SCK, MOSI = 13, 21, 20    # backlight potentiometer
START_LEVEL = 80              # brightness at start, 0-100

pi = pigpio.pi()
if not pi.connected:
    sys.exit("pigpiod service is not running")
for pin, state in ((CS, 1), (SCK, 0), (MOSI, 0)):
    pi.set_mode(pin, pigpio.OUTPUT)
    pi.write(pin, state)

def set_level(level):
    frame = 83 + level * 44 // 100    # 0x00 (write wiper 0), value
    pi.write(CS, 0)
    for bit in range(15, -1, -1):
        pi.write(MOSI, (frame >> bit) & 1)
        pi.write(SCK, 1)
        pi.write(SCK, 0)
    pi.write(CS, 1)

app = QApplication(sys.argv)
window = QWidget()
window.setWindowTitle("Riverdi Brightness")
label = QLabel()
slider = QSlider(Qt.Horizontal)
slider.setRange(0, 100)

def on_change(level):
    set_level(level)
    label.setText(f"Brightness: {level} %")

slider.valueChanged.connect(on change)
layout = QVBoxLayout(window)
layout.addWidget(label)
layout.addWidget(slider)
slider.setValue(START_LEVEL)
on_change(START_LEVEL)
window.resize(360, 90)
window.show()
sys.exit(app.exec_())
EOF
```

3. Make the application executable:

```
sudo chmod 755 /usr/local/bin/riverdi-brightness.py
```

4. Create the autostart entry:

```
mkdir -p ~/.config/autostart
cat > ~/.config/autostart/riverdi-brightness.desktop << 'EOF'
[Desktop Entry]
Type=Application
Name=Riverdi Brightness Control
Exec=/usr/local/bin/riverdi-brightness.py
EOF
```

5. Configure the module to start the desktop with automatic login:

```
sudo raspi-config nonint do_boot_behaviour B4
```

5.9 Step 8 – Reboot and Verify

1. Reboot the module:

```
sudo reboot
```

2. Wait until the desktop is shown on the LCD and the Riverdi Brightness window opens.
3. Carry out the checks in Section 6.

5.10 Series Deployment

All steps after writing the image consist of commands and file edits and can be combined into a provisioning script. The Riverdi settings block from Step 5 can also be added to `config.txt` on the boot partition of a freshly written eMMC or microSD card on the host PC before the first start.

6. VERIFICATION

6.1 System Information

```
tr -d '\0' < /proc/device-tree/model; echo
head -n1 /etc/rpi-issue
uname -r
vcgencmd get_throttled
```

Expected: the model string contains `Compute Module 4`, and `get_throttled` returns `throttled=0x0`.

6.2 Display

```
for c in /sys/class/drm/card*-HDMI-*/; do echo "$(basename "$c"): $(cat "$c/status")"; done
tr ' ' '\n' < /proc/cmdline | grep '^video='
```

CHECK	EXPECTED RESULT
Connector status	`card0-HDMI-A-1: connected`
Kernel command line	`video=HDMI-A-1:1280x800M@60`

6.3 LCD Control Signals

```
pinctrl get 22,23
```

Expected: GPIO22 and GPIO23 are configured as outputs (`op`) with level high (`hi`).

6.4 Touch

```
lsusb | grep 222a
grep -A4 'ILITEK' /proc/bus/input/devices
```

Expected: `ID 222a:0001 ILI Technology Corp. Multi-Touch Screen` and the input device `ILITEK ILITEK-TOUCH`.

For an interactive test, install and start evtest, select the ILITEK device and touch the screen. `ABS_MT_POSITION_X` and `ABS_MT_POSITION_Y` events are reported:

```
sudo apt install -y evtest
sudo evtest
```

6.5 Brightness

```
systemctl is-active pigpiod
```

Expected: `active`. Move the slider in the brightness control window; the backlight brightness follows the slider position.

6.6 I²C Devices (Optional)

```
sudo raspi-config nonint do_i2c 0
sudo apt install -y i2c-tools
i2cdetect -y 1
```

Expected addresses: 0x20, 0x2b, 0x2f, 0x41, 0x44, 0x51 and 0x62.

7. APPLICATION BRIGHTNESS CONTROL

Applications can set the brightness directly instead of using the brightness control application. In this case, remove the autostart entry created in Step 7 so that only one application writes the brightness value. The function `set_level()` in Step 7 is a reference implementation of the frame below.

PARAMETER	VALUE
Chip select	GPIO13, active low
Clock	GPIO21, idle low
Data	GPIO20
Format	SPI mode 0, MSB first, 16 bits per write
Frame	`0x00` followed by the brightness value
Brightness value	83 + level × 44 / 100, for level 0 – 100
Example	Level 80 → value 118 → frame `0x00 0x76`