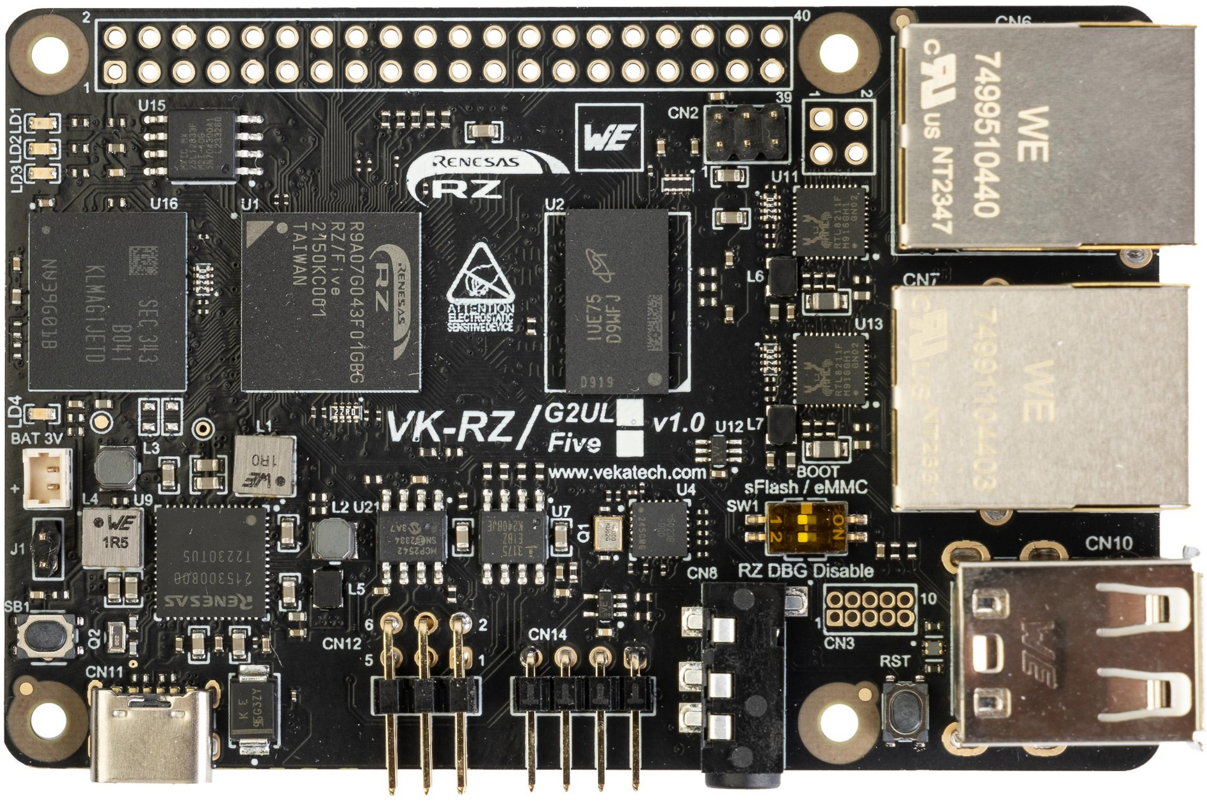
VK-RZ/G2UL How To

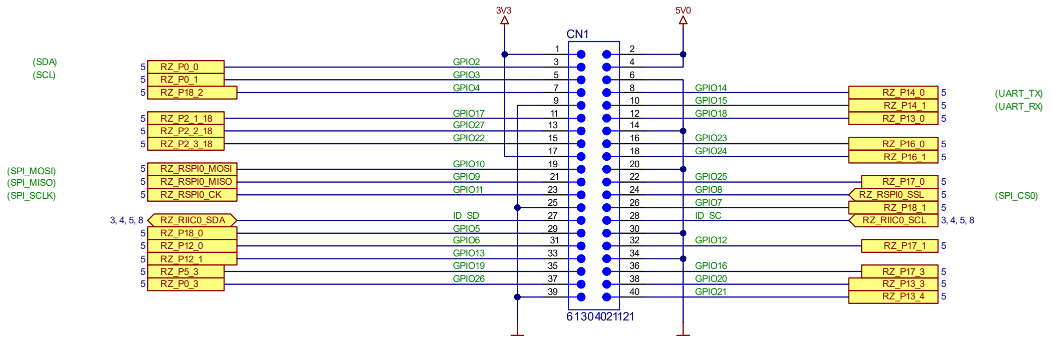

VK-RZ/G2UL v1.0 Board

Content:

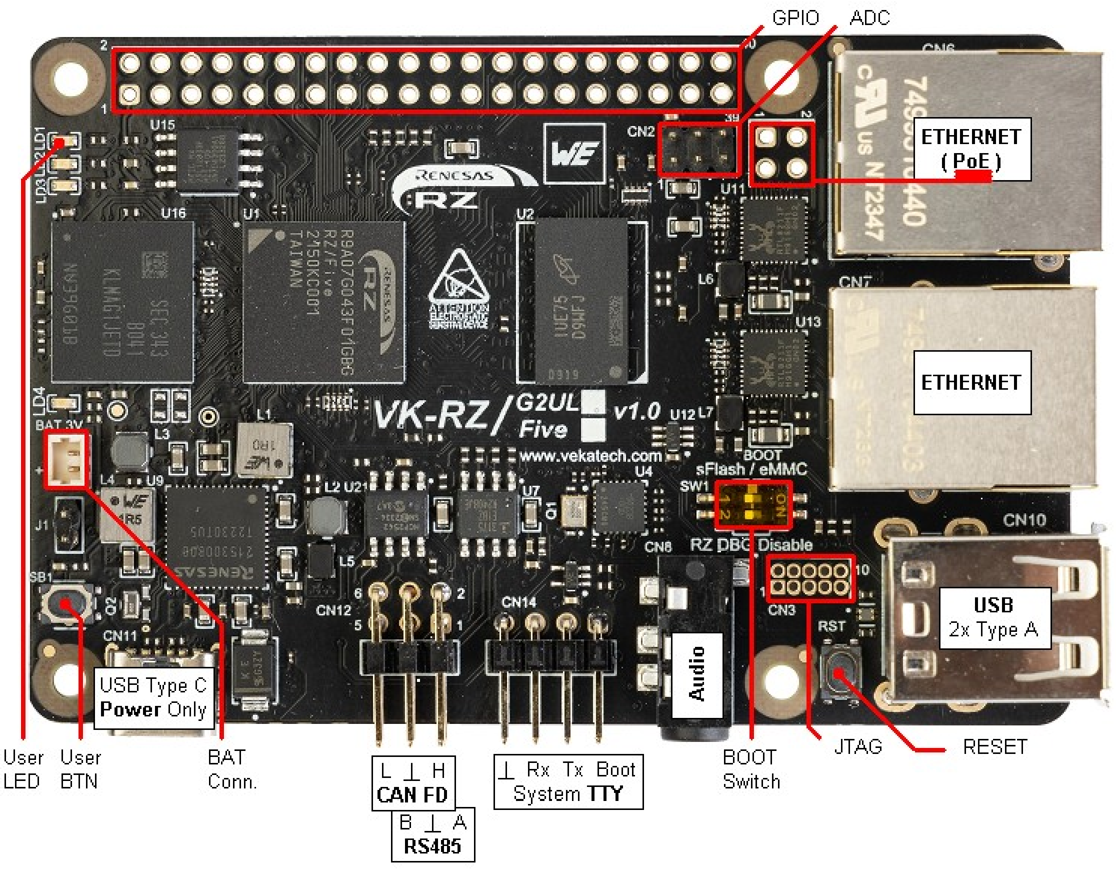
1. Introduction

VK-RZ/G2UL is industrial oriented board, compatible with Raspberry Pi 4 shields. It is based on Renesas **R9A07G043U11GBG**, **ARM Cortex-A55 + Cortex-M33 MCU**. The main purpose of this manual is to show how to get started with the board.

1.1 Connectors

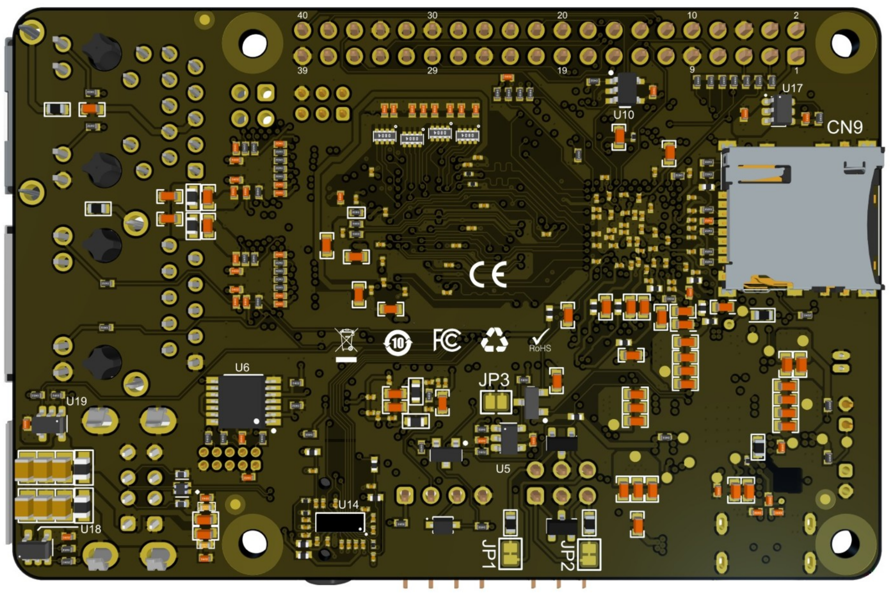


GPIO



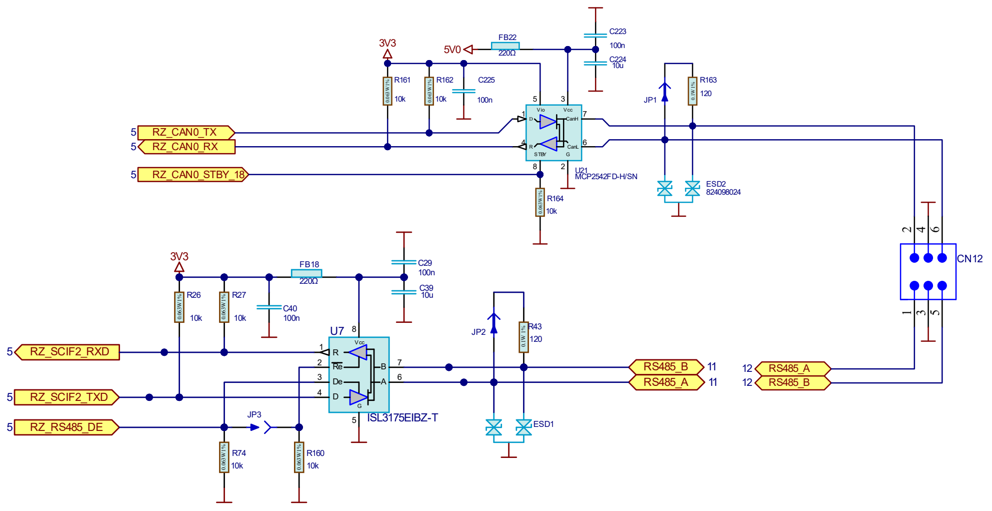
Connectors & Signals

1.2 Signals


Joint Solderpads

Some of the pads below are connected by default other are disconnected. You can change them to suit your needs:

- **JP 1** (short) → Connects **120Ω termination** resistor between CAN's **H** & **L** lines.
- **JP 2** (short) → Connects **120Ω termination** resistor between RS485's **A** & **B** lines.
- **JP 3** (open) → Connects **RS485 Transceiver's DE & $\overline{\textbf{RE}}$** signals together.

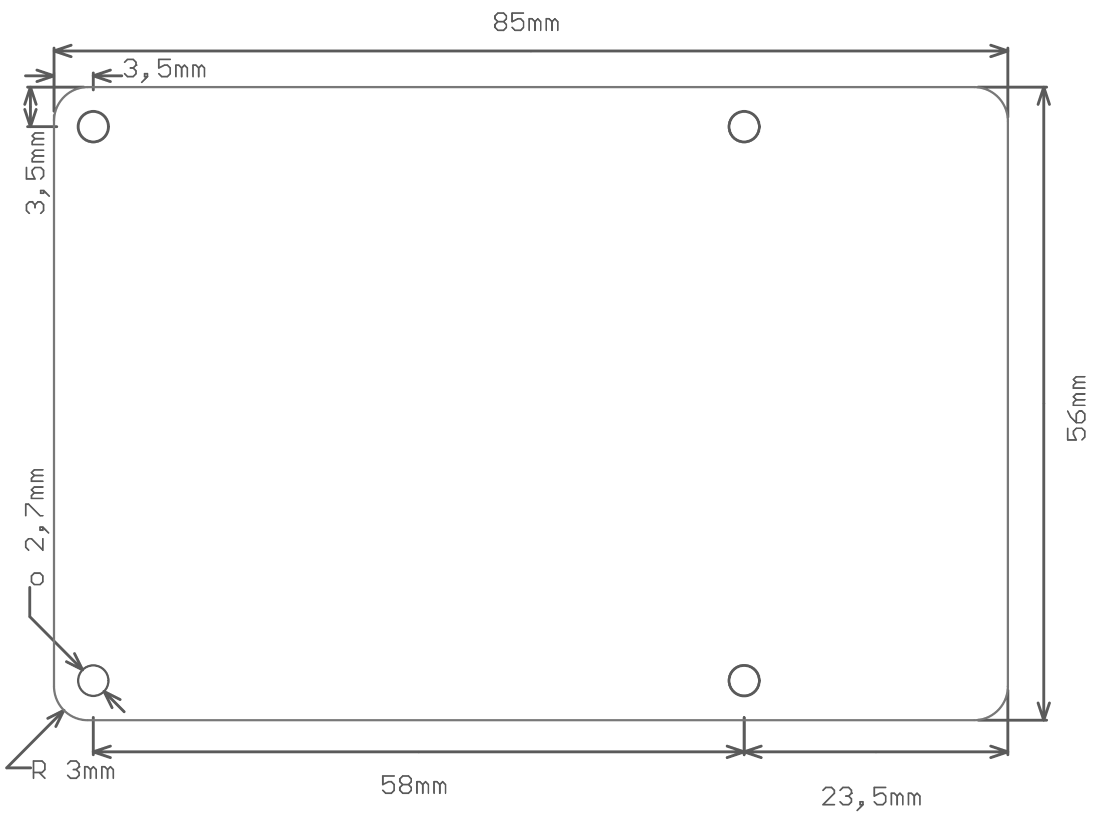


JP1, JP2 & JP3

1.3 3D view

There is available step model of the board. It is convenient if you want to make a case or embed it into a bigger device, so mechanical data can be exported in great detail.

1.4 Dimensions



Board size & Mounting Holes

2. Power Up

The board can be powered with **5 V** from 2 sources:

- USB Type C (Power Only)
- 40pin Extension header

In idle state (without plugged periphery) the board consumes ~ **300 mA**. The consumption however, can jump up to ~ **1,5 A** (if both USB devices uses their full sourcing potential i.e 0,5A each).

3. Install U-boot (v2021.10)

You can get the **U-boot** firmware from our site or compile it yourself from our repository, if you prefer manually building it from scratch.

- Connect **VK-RZ/G2UL** to the PC (through **UART to USB** cable converter) & see what COM port is assigned by the OS in the Device Manager.
- Connect **5V** (USB VBUS) to the boot pin of **CN14** so the board will boot from the **SCIF0**.
- Download **vkPyFlasher** rescue tool and launch it.
- Download the U-boot **firmware** and unpack it in: **vkPyFlasher/images**.

3.1 into SPI Flash

- Run the following commands:

```
cd vkPyFlasher.
flash_boot.py --board=vkrzg2ul --serial_port=COM<n> --qspi.
```

- Press **reset** button on the **VK-RZ/G2UL**
- Wait the flashing to complete.

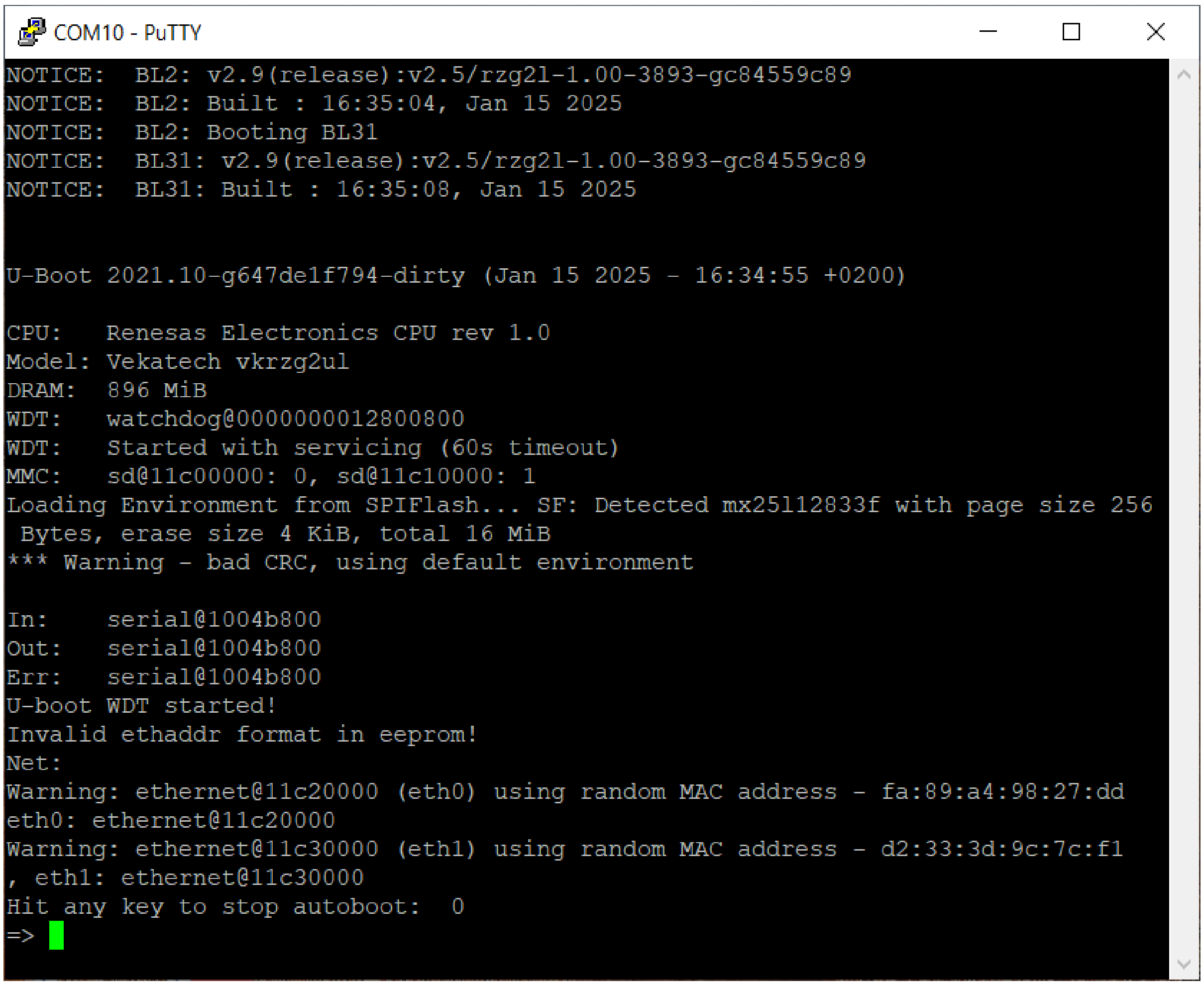
3.2 into eMMC SSD

- Run the following commands:

```
cd vkPyFlasher.
flash_boot.py --board=vkrzg2ul --serial_port=COM<n>.
```

- Press **reset** button on the **VK-RZ/G2UL**
- Wait the flashing to complete.

After download is ready, remove **5V** from boot pin of **CN14** and make sure switch **SW1:1** is **ON** (if you want to boot from **eMMC**) or **OFF** (if you want to boot from **SPIBSC**) and press **reset** => you should now be able to see the boot log of the U-boot.



COM10 - PuTTY

```
NOTICE:  BL2: v2.9(release):v2.5/rzg2l-1.00-3893-gc84559c89
NOTICE:  BL2: Built : 16:35:04, Jan 15 2025
NOTICE:  BL2: Booting BL31
NOTICE:  BL31: v2.9(release):v2.5/rzg2l-1.00-3893-gc84559c89
NOTICE:  BL31: Built : 16:35:08, Jan 15 2025


U-Boot 2021.10-g647de1f794-dirty (Jan 15 2025 - 16:34:55 +0200)

CPU:   Renesas Electronics CPU rev 1.0
Model: Vekatech vkrzg2ul
DRAM:  896 MiB
WDT:   watchdog@0000000012800800
WDT:   Started with servicing (60s timeout)
MMC:   sd@11c00000: 0, sd@11c10000: 1
Loading Environment from SPIFlash... SF: Detected mx25l12833f with page size 256
 Bytes, erase size 4 KiB, total 16 MiB
*** Warning - bad CRC, using default environment

In:    serial@1004b800
Out:   serial@1004b800
Err:   serial@1004b800
U-boot WDT started!
Invalid ethaddr format in eeprom!
Net:
Warning: ethernet@11c20000 (eth0) using random MAC address - fa:89:a4:98:27:dd
eth0: ethernet@11c20000
Warning: ethernet@11c30000 (eth1) using random MAC address - d2:33:3d:9c:7c:f1
, eth1: ethernet@11c30000
Hit any key to stop autoboot:  0
=>
```

U-boot log

4. Boot logic

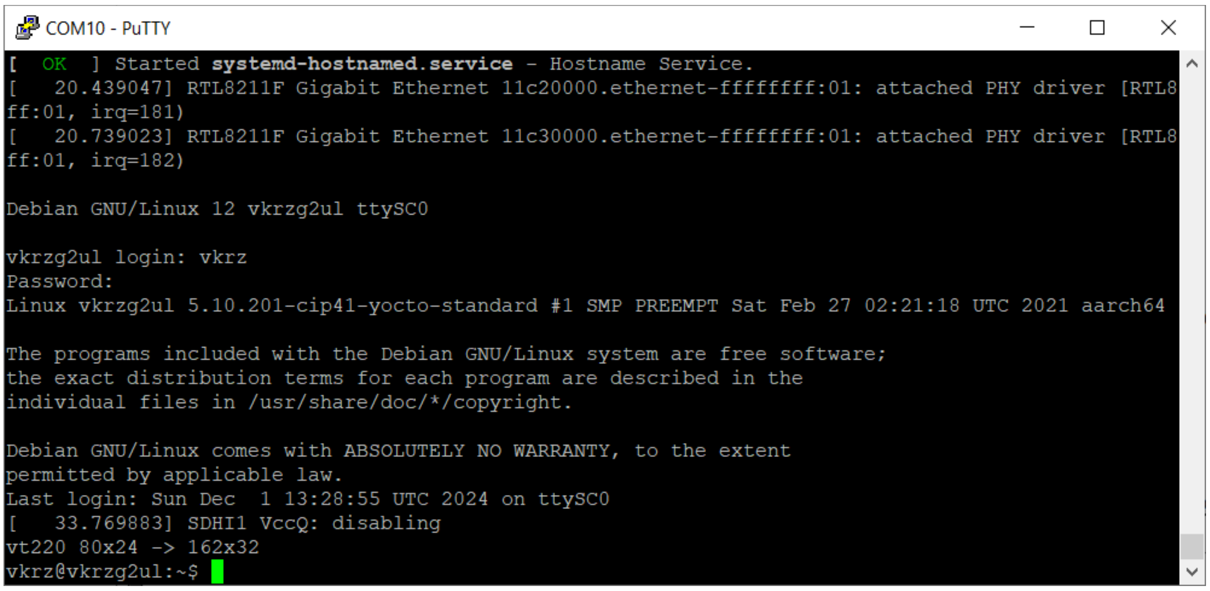
Understanding boot logic is crucial. As you see there are **2** copies of **U-boot** (1 in eMMC & 1 in SPI). Environment parameters of the ones are slightly different from the others. They are configured in such way, that when **SW1** is set to boot from eMMC (**1:ON**) it's U-boot searches Linux image in the same that **eMMC**, but when SW1 is set to boot from SPI (**1:OFF**) it's U-boot searches Linux image in **µSD** card. That's why default U-boot environment parameters differs on purpose and you don't need to edit them every time you want to boot from one or other media (you just set SW1). That actually explains why booting from SPI, leads to booting from µSD. You can always change those parameters & boot from wherever you want.

5. Install Linux (Kernel v5.10.xxx)

Now that you have 2 workable U-boot instances, up & running, it is time to load something bigger. You can get Debian Linux image from our site or compile Yocto yourself from our repository, (if you prefer building it from scratch)::

5.1 into SD card → Debian v12.x (Bookworm)

- Get a **µSD** card with **min 4 G** capacity and plug it into the PC.
- Download tool, named Rufus from here.
- Unzip **debian-bookworm-vkrzg2ul.img.xz**.
- Launch Rufus, for **Device** select µSD card drive, for **boot section** select desired Linux image file (in this case **debian-bookworm-vkrzg2ul.img**).
- Hit **START** & wait completion (Status **READY**), and eject µSD card from the PC.
- Plug µSD card into VK-RZ/G2UL's holder, set **SW1** to (1:OFF) & press reset.
- You should now be able to see login screen, use user: **`vkrz`** & password: **`vkrzg2ul`**.



COM10 - PuTTY

```
[  OK  ] Started systemd-hostnamed.service - Hostname Service.
[   20.439047] RTL8211F Gigabit Ethernet 11c20000.ethernet-fffffff:01: attached PHY driver [RTL8
ff:01, irq=181)
[   20.739023] RTL8211F Gigabit Ethernet 11c30000.ethernet-fffffff:01: attached PHY driver [RTL8
ff:01, irq=182)

Debian GNU/Linux 12 vkrzg2ul ttySC0

vkrzg2ul login: vkrz
Password:
Linux vkrzg2ul 5.10.201-cip41-yocto-standard #1 SMP PREEMPT Sat Feb 27 02:21:18 UTC 2021 aarch64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Dec  1 13:28:55 UTC 2024 on ttySC0
[   33.769883] SDHI1 VccQ: disabling
vt220 80x24 -> 162x32
vkrz@vkrzg2ul:~$
```

Debian 12 login screen

- You can check for available package updates: `sudo apt-get update`.

If there are available upgrades such as in this case:

```
xx packages can be upgraded. Run 'apt list --upgradable' to see them.
```

- You can install them: `sudo apt-get upgrade`.
- If you want to extend the root partition to use the full capacity of the µSD:

`sudo growpart /dev/mmcblk1 2 && sudo resize2fs /dev/mmcblk1p2`.

- If you want to extend the root partition to a fixed size and form a new separate partition you should first install partition editor `sudo apt-get install parted`.

Extend the root partition to **N** GB size: `sudo parted /dev/mmcblk1 resizepart 2 NG`.

Fix filesystem block size: `sudo resize2fs /dev/mmcblk1p2`.

Make new partition: `sudo parted /dev/mmcblk1 "mkpart primary fat32 NG -1"`.

Format the new partition: `sudo mkfs.vfat -F 32 /dev/mmcblk1p3`.

5.2 into eMMC SSD → Yocto v3.1.xx (Dunfell)

Get the **vkPyFlasher** prepared:

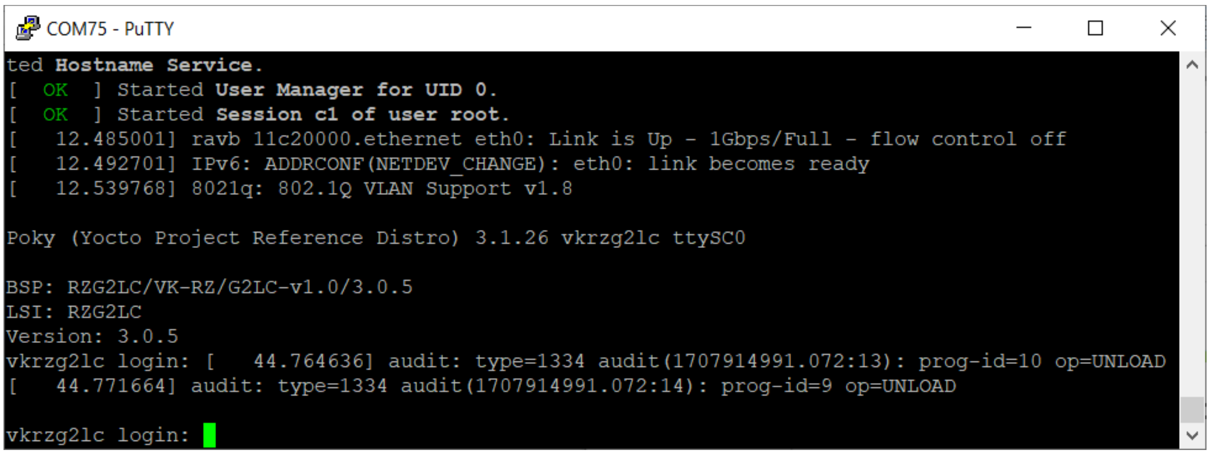
- Connect **VK-RZ/G2UL** to the PC (through **UART to USB** cable) & see what COM port is assigned by the OS in the Device Manager.
- Connect **VK-RZ/G2UL** to the PC (through the **Ethernet** port) via switch/router etc.
- Remove any voltage from boot pin of **CN14**, and set **SW1:1** ON to boot from the **eMMC**.
- Download the desired Linux image (Yocto / Debian), and place it at the location: **vkPyFlasher/images/vkrzg2ul**.

Flash the image:

- `cd vkPyFlasher`.
- In case you want **Yocto**:
  `flash_img.py --board=vkrzg2ul --serial_port=COM<n> --image_rootfs=core-image-bsp-vkrzg2ul.simg --udp`.
- In case you want **Debian**:
  `flash_img.py --board=vkrzg2ul --serial_port=COM<n> --image_rootfs=debian-bookworm-vkrzg2ul.simg --udp`.
- Press **reset** button on the **VK-RZ/G2UL**.
- Wait the flashing to complete.
- Open the COM port assigned by the OS for the VK-RZG2UL with (115200|B|N|1) settings.
- Press **reset** once again.
- Login to Yocto (user: **root**) or Debian (user: **vkrz** & password: **vkrzg2ul**).
- In case of Debian, you can setup the partition on the fly:
  → Extend the root partition to use the full capacity of the eMMC:
  `sudo growpart /dev/mmcblk0 2 && sudo resize2fs /dev/mmcblk0p2`.
- In case of Yocto, you can setup the partition on the fly too, but using fdisk.

`echo -e "d\n2\nn\np\n2\n<Start Sector>\n\nN\nw" | fdisk /dev/mmcblk0`.

`resize2fs /dev/mmcblk0p2`.



```
ted Hostname Service.
[  OK  ] Started User Manager for UID 0.
[  OK  ] Started Session c1 of user root.
[   12.485001] ravb 11c20000.ethernet eth0: Link is Up - 1Gbps/Full - flow control off
[   12.492701] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
[   12.539768] 8021q: 802.1Q VLAN Support v1.8

Poky (Yocto Project Reference Distro) 3.1.26 vkrzg2lc ttySC0

BSP: RZG2LC/VK-RZ/G2LC-v1.0/3.0.5
LSI: RZG2LC
Version: 3.0.5
vkrzg2lc login: [   44.764636] audit: type=1334 audit(1707914991.072:13): prog-id=10 op=UNLOAD
[   44.771664] audit: type=1334 audit(1707914991.072:14): prog-id=9 op=UNLOAD

vkrzg2lc login:
```

Yocto Login screen

6. Launch the installed Linux images

Thanks to the convenient boot logic, launching is easy, it depends on correct setting of **SW1**

- To boot Linux from µSD: set the switch to **SPIBSC (1:OFF)**.
- To boot Linux from eMMC: set the switch to **eMMC (1:ON)**.
- To boot Linux from Ethernet: set the switch to **SPIBSC** and make sure µSD slot is Empty. When U-boot can't find SD card, it will try to boot from its **serverip**, **tftpdir** & **netrootfs** environment parameters. (serverip is the address, where U-boot will look for NFS & TFTP servers, tftpdir shows where the boot directory is on the server and netrootfs → where root directory is on the server) Same server can be used to boot multiple boards, every board can have its own boot and root directories and that's why tftpdir and netrootfs parameters should be filled with the correct paths on the server.

7. Apply Overlays

Overlays are way to tell Linux to use or not a given periphery. They can even point a specific hardware for a same periphery and form configurations for different version of a board. For example in version 1, the board can use **Realtek** audio driver, in other **someone's else**, in third no audio at all. Management of the overlays is done through **uEnv.txt**, located in **/boot** directory of the particular linux image. This file is analyzed during boot of the Linux image, so every change in that file takes effect after boot. To apply overlay, you need to line it up in the following row:

```
fdt_extra_overlays=vkrz-audio.dtbo
```

All available overlays that can be applied are listed in **/boot/overlays**, but simultaneously only these can be activated at the same time.

- vkrz-audio.dtbo
- vkrz-cm33.dtbo
- vkrz-exp-i2c2_100.dtbo / vkrz-exp-i2c2_400.dtbo
- vkrz-exp-i2c3_100.dtbo / vkrz-exp-i2c3_400.dtbo
- vkrz-exp-scif1.dtbo / vkrz-exp-scfi1_rs485.dtbo
- vkrz-exp-scif3.dtbo
- vkrz-exp-spi0.dtbo
- vkrz-udma.dtbo

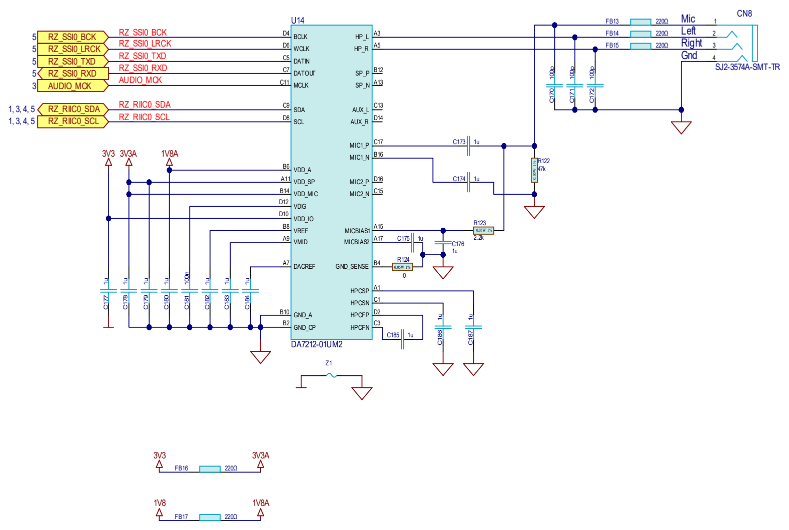
8. Use various periphery in Linux

To demonstrate using different periphery, we are going to use the Debian image, for other images (Yocto for example) the commands could be not available or could be slightly different, so don't worry if some of them do not work, just google how to do it for your particular distribution.

8.1 Audio

Make sure you are applied **vkrz-audio.dtbo** overlay in **/boot/uEnv.txt**, i.e. you should see it is included in the following row:

```
fdt_extra_overlays=vkrz-audio.dtbo
```

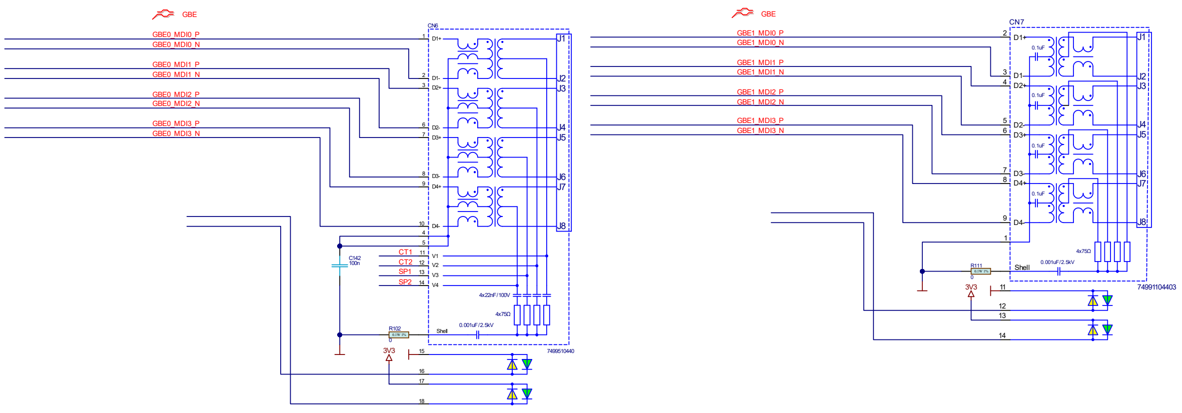


3,5mm CTIA Mini Jack

- Plug headphones in the jack & type: `aplay /usr/share/sounds/alsa/Noise.wav`.
- You should now be able to hear white noise sample. If for some reason you can't here anything, open alsamixer and check if **Card** (in the upper left corner) is **audio-da7212**. If it is different, press **F6** (Select sound card) and choose **audio-da7212**, also find **Mixout Left DAC Left** & **Mixout Right DAC Right** and make sure they are not **Off**.

8.2 Ethernet

Ethernet is enabled by default and does not need overlay. Plug the **cat** cable into any of the **RJ45** connectors and in a couple of seconds the board should get an **IP** from the local **DHCP** server.

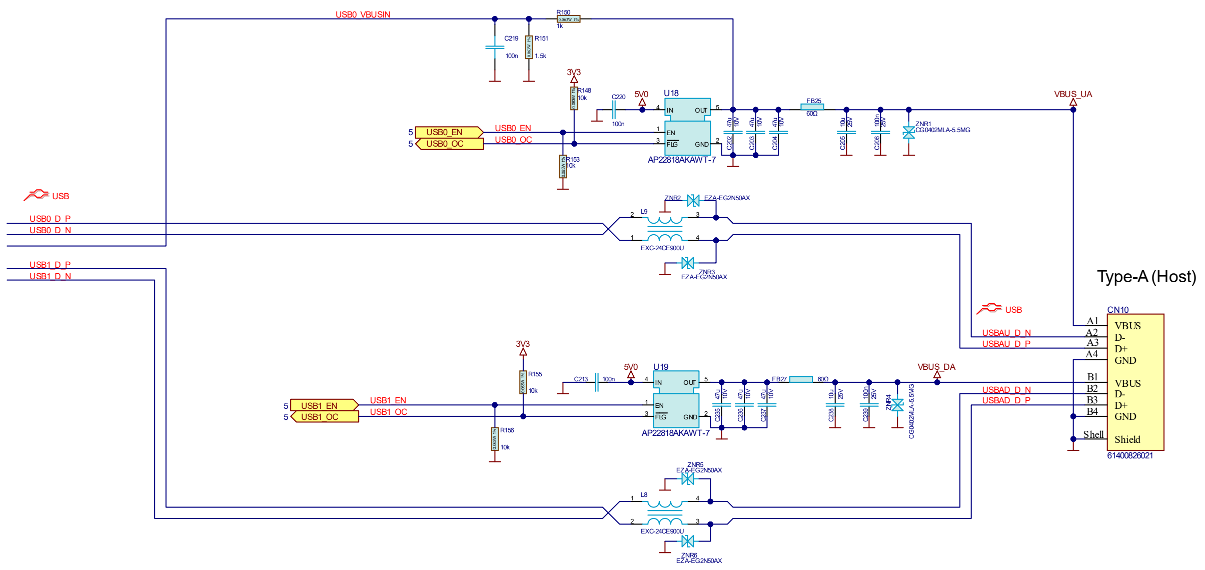


RJ45 connectors

- You can check what IP is assigned: `sudo ifconfig`.
- You can probe if Host is reachable: `ping vekatech.com`.

8.3 USB

USB is also enabled by default and does not need overlay. To test it, plug a USB device such as: Mouse, Keyboard, USB Flash & whatever other USB device you can think of.



USB Type A connectors

- Mouse: It is Plug & Play device , you don't need to do anything.
- Keyboard: Also Plug & Play device, you don't need to do anything.
- Flash Drive: most of the latest Linux distributions mounts the drive automatically, (Debian 12 is not an exception), so it is again another Plug & Play device.

- GSM Modem: Also Plug & Play device, but to set a connection it needs some setup:

→ Make sure `sudo apt-get install network-manager modemmanager` are installed

→ Start their services: `systemctl start NetworkManager ModemManager`.

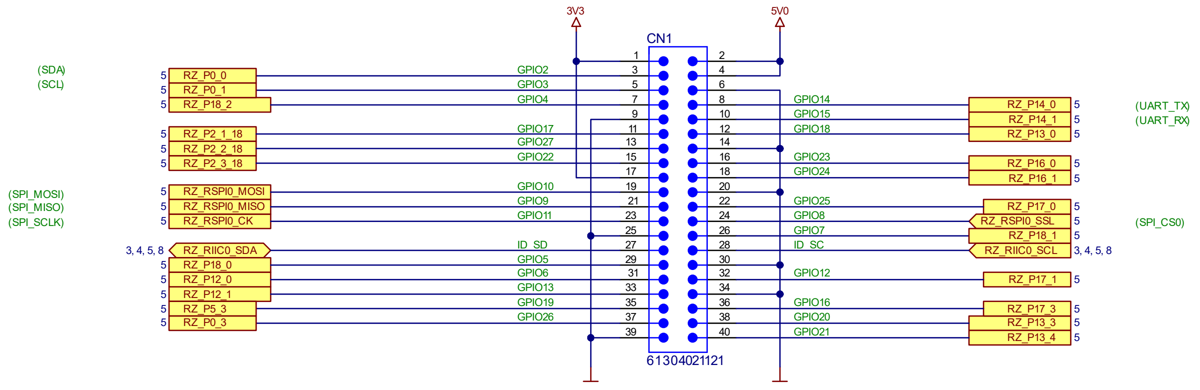
→ If autostart on boot is a must: `systemctl enable NetworkManager ModemManager`.

→ Configure GSM connection : `sudo nmcli connection add type gsm ifname '*' con-name '<name>' apn '<AP>' connection.autoconnect yes`.

→ A new network interface, type **gsm** should appear: `nmcli device status`.

8.4 GPIO

GPIOs are enabled by default and do not need overlay.



40 pin Extension Header

Pay attention that **RZ_P2_1_18, RZ_P2_2_18 & RZ_P2_3_18** are on the **Low Voltage Power Domain** (i.e. their logical **HI** levels are **1,8V**) and take that in consideration when using these GPIOs.

- You have to tell the system which pin you want to manipulate. Let's say the pin is "**Pport_pin**". You have to calculate the internal pin number: (**port** x **8**) + **pin** + **120**, if **P13_0** is desired to be controlled, the internal number is: (**13** x 8) + **0** + 120 = **224**.
  The way to tell the system is with these commands: `sudo su`
  `echo 224 > /sys/class/gpio/export`.
- If you want to get the value of **P13_0** pin, you can type this:
  `echo in > /sys/class/gpio/P13_0/direction`.
  `cat /sys/class/gpio/P13_0/value`.
- If you want to set **P13_0** pin to **HI**, you can type this:
  `echo out > /sys/class/gpio/P13_0/direction`.
  `echo 1 > /sys/class/gpio/P13_0/value`.
- If you don't want to use the pin anymore you can release the resource:
  `echo 224 > /sys/class/gpio/unexport`.

8.5 SPI

Make sure you are applied **vkrz-exp-spi0.dtbo** overlay in **/boot/uEnv.txt**.

```
fdt_extra_overlays=vkrz-exp-spi0.dtbo
```

SPI[0] pins go to Pin19 (**MOSI**), Pin21 (**MISO**), Pin23 (**SCLK**), Pin24 (**CS0**) of 40pin ext. connector.

- Install SPI software: `sudo apt-get install spi-tools`.
- See the default config of SPI bus: `sudo spi-config -d /dev/spidev0.0 -q`.
- Set the config you need, use `spi-tools –help` to see what can be configured.
- Short MOSI & MISO together (Pins 19 & 21), so a Loopback test can be run.
- Send some data and see if it is received properly: `echo -n -e "1234567890" | sudo spi-pipe -d /dev/spidev0.0 -s 10000000 | hexdump`.
- If everything is OK you should see the digits: `3231 3433 3635 3837 3039`.
- Remove the short and try again sending the digits: `echo -n -e "1234567890" | sudo spi-pipe -d /dev/spidev0.0 -s 10000000 | hexdump`.
- Now you should see a complete flat line (Vcc): `ffff ffff ffff ffff ffff`.

8.6 I2C

Make sure you are applied **vkrz-exp-i2c2_100.dtbo** or **vkrz-exp-i2c2_400.dtbo** and/or **vkrz-exp-i2c3_100.dtbo** or **vkrz-exp-i2c3_400.dtbo** overlay in **/boot/uEnv.txt**.

```
fdt_extra_overlays=vkrz-exp-i2c2_100.dtbo vkrz-exp-i2c3_100.dtbo.
```

The suffix **_100** or **_400** indicates the speed on which the I2C works (i.e. 100 or 400 kHz)

I^{2}C[2] pins go to Pin35 (**SDA**) & Pin37 (**SCL**) of 40pin extension connector.
I^{2}C[3] pins go to Pin3 (**SDA**) & Pin5 (**SCL**) of 40pin extension connector.

- Install I^{2}C software: `sudo apt-get install i2c-tools`.
- Now we can see all I^{2}C devices on the system bus 0: `sudo i2cdetect -y -a -r 0`.

```
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- 1a -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- UU -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
```

I^{2}C[0] pins also goes to the 40pin extension connector → Pin27 (**SDA**) & Pin28 (**SCL**)

- We can also see all I²C devices on the system bus 1: `sudo i2cdetect -y -a -r 1`.

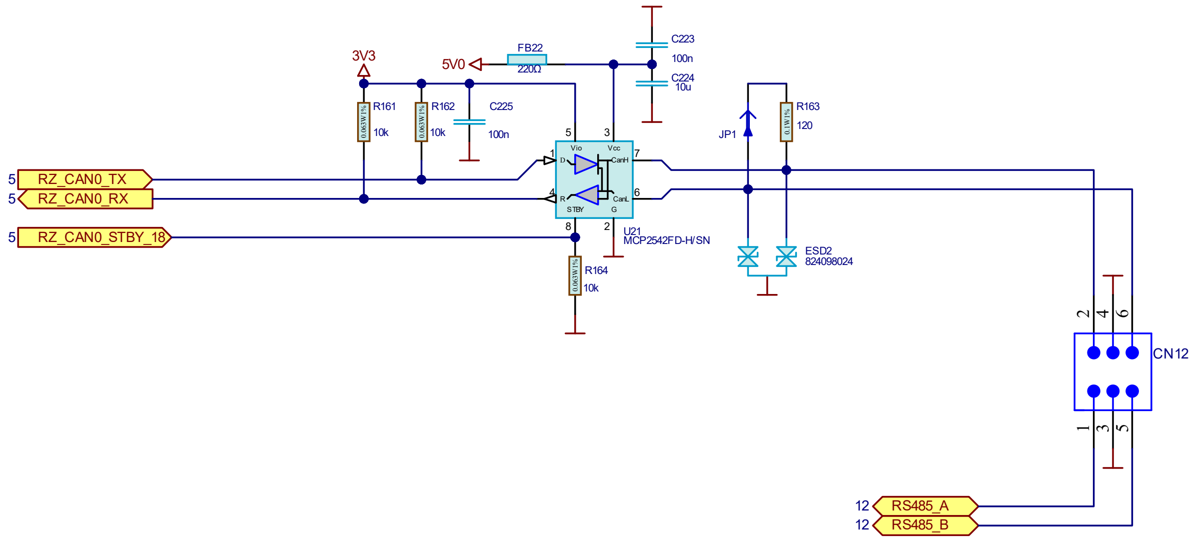
```
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: UU UU UU UU 54 55 56 57 -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
```

I²C[1] is internal bus and does not go to the 40pin extension connector, however this of course doesn't mean we can't access the device there:

- Read one byte from addr 0x00 of E²PROM: `sudo i2cget -y 1 0x54 0x00 b`.
- Write byte 0x55 to addr 0xF0 of E²PROM: `sudo i2cset -y 1 0x54 0xF0 0x55 b`.

8.7 CAN

CAN is enabled by default and do not need overlay.



CAN

Use the CAN module:

- Install CAN software: `sudo apt-get install can-utils`.
- Turn off the can module: `sudo ip link set can0 down`.
- Set the config you need: `sudo ip link set can0 type can bitrate 2000000 dbitrate 2000000 fd on`.
- Turn on the can module: `sudo ip link set can0 up`.
- Here a Loopback test is possible if only a special mode exist in the can module, but this is not the case here, so you will need a second can device to talk to.
- If you want to receive messages type: `candump can0`.
- If you want to send message type: `cansend can0 123#0102030405060708`.
- Use `cansend -help` for more info for the format of the can messages (in this case, 11bit address mode is used → [addr: **123**] & full msg len [data: **0102030405060708**])

8.8 UART

Make sure you are applied **vkrz-exp-scfi1.dtbo** or/and **vkrz-exp-scfi3.dtbo** overlay in **/boot/uEnv.txt**.

```
fdt_extra_overlays=vkrz-exp-scfi1.dtbo vkrz-exp-scfi3.dtbo
```

UART[1] pins go to Pin8 (**TX**), Pin10 (**RX**) of 40pin extension connector.

UART[3] pins go to Pin31 (**TX**), Pin33 (**RX**) of 40pin extension connector.

- Install port software: `sudo apt-get install screen`.
- Short TX & RX together (Pins 8 & 10) or (Pins 31 & 33) so a Loopback test can be run.
- Execute: `screen /dev/ttySC1` or `screen /dev/ttySC3`.
- Type something and you should now be able to see what you are typing.
- If you remove the short, you won't see what you are typing.
- Press **Ctrl+A** and then **K** to exit, (you may need to apply with **y**)

8.9 RS485

RS485 is enabled by default and do not need overlay, but if you want to use **second** RS485 make sure you are applied **vkrz-exp-scfi1_rs485.dtbo** overlay in **/boot/uEnv.txt**.

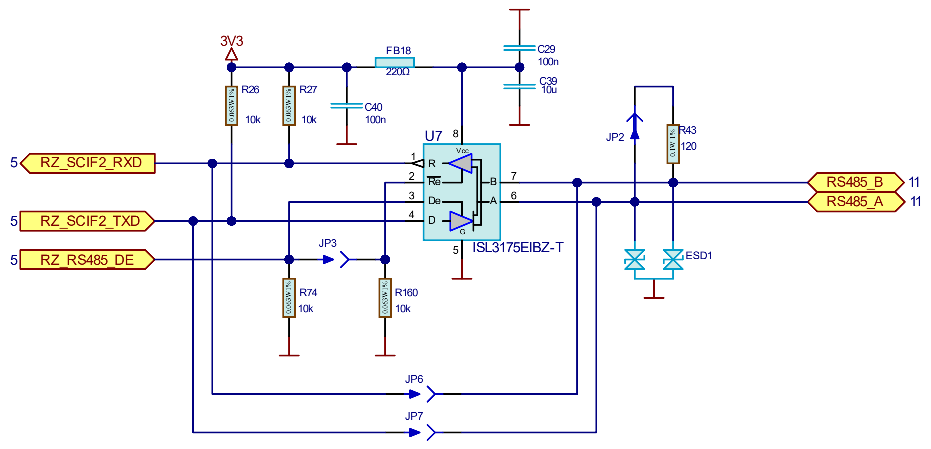
```
fdt_extra_overlays=vkrz-exp-scfi1_rs485.dtbo
```

UART[1] pins go to:

- Pin8 (**TX**), Pin10 (**RX**), Pin18 (**DE**) of 40pin extension connector.

For UART[**1**] you will need **external RS485 Transceiver** connected to Pin**8** Pin**10** and Pin**18**.

UART[2] you can use it directly:



RS485

- Install port software: `sudo apt-get install screen`.
- Connect a second RS485 device to connector **CN12** → Pin1 (**A**) Pin3 **GND** Pin5 (**B**).
- Execute: `screen /dev/ttySC2`.
- Type something and you should now be able to see what you are typed on the **second** RS485 device. Type something on the second RS485 device and you should be able to see what you are typed on the **screen**.
- Press **Ctrl+A** and then **K** to exit, (you may need to apply with **y**)

Revision overview list

Revision number	Description changes
0.1	Initial

Vekatech Ltd.

63, Nestor Abadzhiev st.
4023 Plovdiv
Bulgaria
Tel.: +359 (0) 32 262362
info@vekatech.com

www.vekatech.com