VE2302 Development Kit
Getting Started Guide

Version 1.0

LIT# VE2302-DK-GSG-V1P0

Contents

Figures

1 Document Control

Document Version: 1.0

Document Date: 9 January 2026

2 Version History

Version	Date	Comment
1.0	01/09/2026	Initial Release

3 Pertinent Info

The VE2302 Development Kit is a solution that incorporates the Tria VE2302 System-On-Module (SOM) based on Versal AI Edge APSoC and a carrier board targeted for broad use in many applications:

- Offers a Versal AI Edge based Development Kit in Commercial (0ºC to 70ºC) temperature grade for engineers to adopt in development, proof-of-concept, and production projects.
- Combines programmable logic designs with dual-core ARM® Cortex®-A72 MPCore™ and dual-core Arm Cortex-R5F MPCore in a convenient and expandable board.
- Allows expansion to a variety of peripherals through the HSIO (High-Speed IO) and MIPI 22-pin expansion connectors.
- Designers can create or evaluate designs for both the Versal Processing Subsystem (PS) and / or the Programmable Logic (PL) fabric.

  The following figure is a high-level block diagram of the VE2302 Development Kit, and the peripherals attached to the Tria VE2302 SOM.

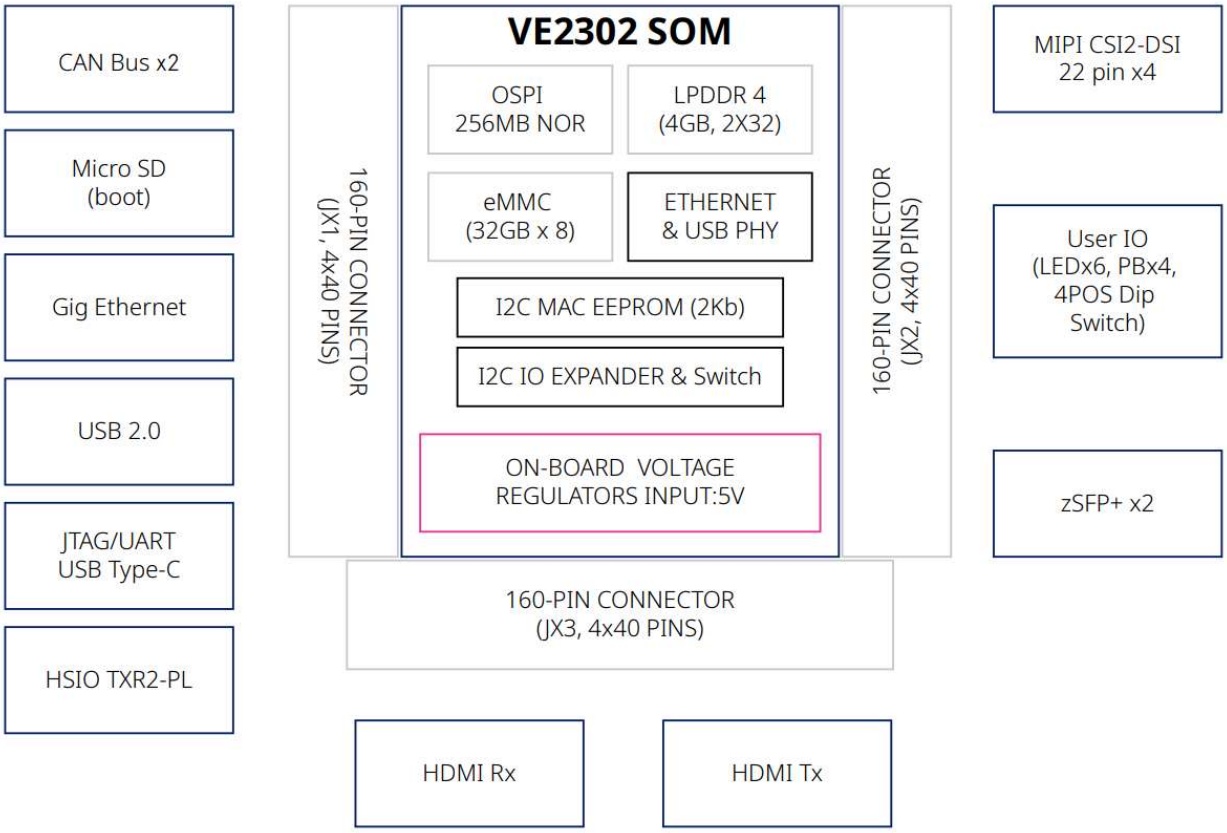

Figure 1 – Development Kit Block Diagram

The following figure provides an overview of the physical connections, their designators, and relative position on the VE2302 Development Kit.

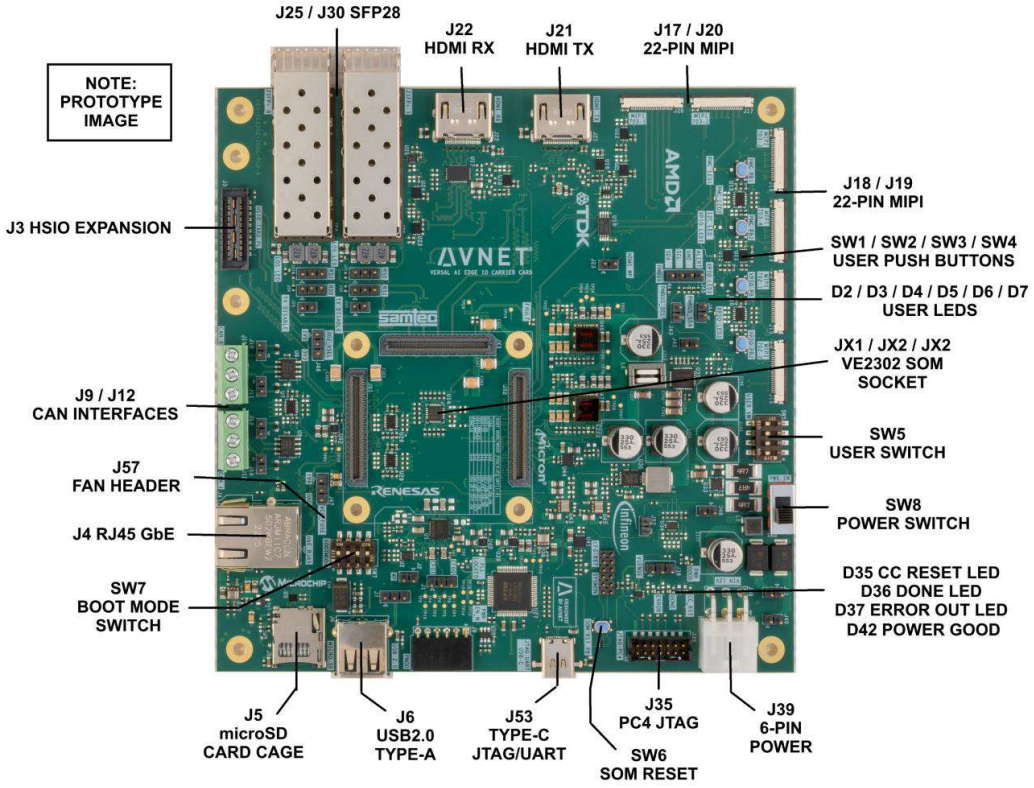

Figure 2 – Interfaces and Connectors

The Getting Started Guide will outline the steps to setup the VE2302 Development Kit hardware. It documents the procedure to run the Out-of-Box design that targets the ARM® Cortex™-A72.

3.1 What's In The Box

The VE2302 Development Kit includes the Tria VE2302 SOM and the Tria Versal AI Edge Carrier Card. The VE2302 Development Kit includes the following in the box:

- Tria VE2302 SOM – Commercial Temperature
- Tria Versal AI Edge Carrier Card
- 12V Power Supply
- Heatsink, Fan, and Mounting Hardware
- Quick Start Card

Customers will need to acquire an appropriate Type-C cable and Ethernet cable to operate the Out-of-Box Design that is programmed into the OSPI of the VE2302 SOM.

3.2 What's On The Web

The VE2302 Development Kit has documentation that is made available to users through the Tria Board product pages and various community support websites. Here are links to the various online content:

- **http://avnet.me/ve2302-dk**
- **http://avnet.me/ve2302-dk-forum**
- **http://avnet.me/board-support-site**

3.3 Available Documentation

- Getting Started Guide
- Hardware User Guide
- Board Definition Files
- Master User Constraint File
- Net Length Report
- Schematics *
- Bill of Materials *
- Mechanical Drawing *
- 3D Model *

* Denotes FAE involvement to gain access to these documents. Contact your local Avnet FAE.

3.4 Tutorials and Reference Designs

Any tutorials and reference designs that are available to targets this platform can be located at the links below.

- Out of Box Design – Part of this Getting Started Guide
- **http://avnet.me/ve2302-dk**

3.5 Trainings and Videos

Any trainings and videos that may be available to target this platform would be located at the links below. At the time of generating this Getting Started Guide there has not been training material published for this platform.

- Community based boards trainings and videos
  - **http://avnet.me/community-boards-training**

4 Getting Started

The functionality of the VE2302 Development Kit is determined by the application booted from the non-volatile memory. In the case of the VE2302 Development Kit there are two options for booting: the VE2302 SOM on-board OSPI and the Versal AI Edge IO Carrier Card SD card interface. For this Getting Started tutorial, the user will utilize the Out-of-Box image pre-programmed on the VE2302 SOM. This program will allow the end user to quickly bring up the board with a network connection and run a webserver application built into a Petalinux reference design.

Reminder: In addition to the items included in the box, you will also need the following to complete the Getting Started tutorial.

- **Type-C Debug Cable for USB-UART Communications**
- **Open Ethernet Port on Host Windows PC**
- **Ethernet Cable**

4.1 Out-of-Box Example Design

The VE2302 Development Kit has an example Petalinux base BSP that includes a webserver used as it's Out-of-Box experience. In this Getting Started tutorial, we will boot the VE2302 Development Kits Petalinux base BSP from OSPI of the Tria VE2302 SOM with the Out-of-Box design.

The Out-of-Box Example Design can be created from scratch by leveraging the available VE2302 Out-of-Box Vivado HW, Petalinux build, and Board Definition Files directories that exist as repositories on the Avnet GITHUB.

- **https://github.com/Avnet/ve2302_oob_hw** - Vivado Project Repository
- **https://github.com/Avnet/petalinux-ve2302-oob** - Petalinux Project Repository
- **https://github.com/Avnet/bdf** - Board Definition Files Repository

4.2 Hardware Setup

There are a few tasks required to ensure the hardware is setup appropriately to run the Out-of-Box Petalinux web server design.

Verify that the appropriate headers are shunted on the Versal AI Edge Carrier Card. The headers should be populated at the factory, but to be thorough an end user should check they have been populated. Not all headers are required to support the Out-Of-Box Design but they are listed here to aid the end-user.

J7 – USB SHIELD - SHUNTED – PINS 2-3
J10 – CAN0 TERMINATION OPTION - SHUNTED – PINS 1-2
J11 – CAN0 TERMINATION OPTION - SHUNTED – PINS 1-2
J13 – CAN1 TERMINATION OPTION - SHUNTED – PINS 1-2
J14 – CAN1 TERMINATION OPTION - SHUNTED – PINS 1-2
J23 – HDMI EEPROM WRITE PROTECT - OPEN – SHUNT NOT PLACED
J26 – zSFP0+ OPTIONS - OPEN – SHUNT NOT PLACED
J27 – zSFP0+ OPTIONS - OPEN – SHUNT NOT PLACED
J28 – zSFP0+ OPTIONS - OPEN – SHUNT NOT PLACED
J31 – zSFP1+ OPTIONS - OPEN – SHUNT NOT PLACED
J32 – zSFP1+ OPTIONS - OPEN – SHUNT NOT PLACED

J33 – zSFP1+ OPTIONS - OPEN – SHUNT NOT PLACED
J37 – JTAG POR OPTION - OPEN – SHUNT NOT PLACED
J38 – SOM RESET TIMING SELECT - SHUNTED – PINS 2-3
J40 – 12V OPTIONAL POWER SUPPLY SUPPORT - OPEN – SHUNT NOT PLACED
J43 – PMBUS SCL TO VERSAL - OPEN – SHUNT NOT PLACED
J44 – PMBUS SDA TO VERSAL - OPEN – SHUNT NOT PLACED
J46 – VCCO_302 BANK VOLTAGE SELECT - SHUNTED – PINS 1-2
J47 – VCC_FUSE PROGRAM SETTING SELECT - SHUNTED – PINS 2-3
J48 – POWER SEQUENCER DEBUG - OPEN – SHUNT NOT PLACED
J54 – CARRIER CARD RESET DEBUG - SHUNTED – PINS 1-2
J56 – POWER SEQUNCE DEBUG - OPEN – SHUNT NOT PLACED
J58 – GTYP CLOCK SELECT – OPEN – SHUNT NOT PLACED
J59 – MIPI DSI 5V POWER INTERFACE - OPEN- SHUNT NOT PLACED

1) The Tria VE2302 SOM should be mounted to VE2302 SOM Slot on the Versal AI Edge Carrier Card with the provided fan and heatsink installed. Please review the Technical Documents section on the product webpage for instructions on assembling and installing the fan and heatsink:

    a. **http://avnet.me/ve2302-dk**

2) The appropriate cables should be attached to the Versal AI Edge Carrier Card. The cables required to run the Out-Of-Box design are as follows:

    - Ethernet Cable should be attached from host PC ethernet interface to the Versal AI Edge Carrier Card RJ45 connector, **J4**.
    - USB Type-C Cable should be attached from host PC USB interface to the Versal AI Edge Carrier Card Type-C USB JTAG/UART connector, **J53**.
    - Provided 12V Power Supply should be attached to the Versal AI Edge Carrier Card 6-pin power connector, **J39**.

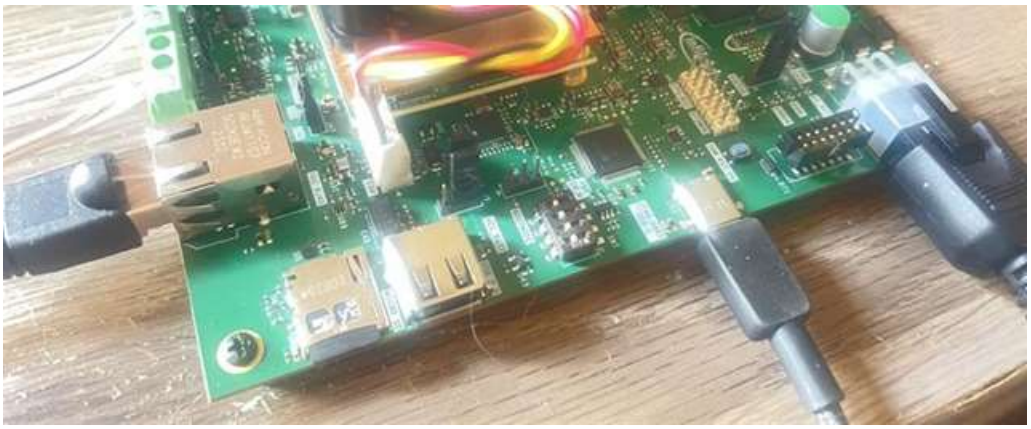

Figure 3 – Cable Connections for OOB Design

3) If not already set, set the Boot Mode Switch, **SW7**, to boot from the OSPI configuration flash. This will allow the Petalinux base BSP design to boot from the OSPI. OSPI mode is set by toggling the Boot Mode Switch to set **SW7[4-1] to OFF-ON-ON-ON** which will set the **MODE[3:0]** pins to **0x1000**.

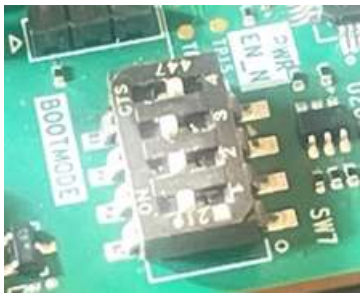

Figure 4 – OSPI Boot Mode Settings

4.3 Terminal Setup

1) A terminal program is required. Tera Term was used in this example which can be downloaded from the Tera Term project. Install Tera Term or another terminal program of your choice to a host Windows PC.

   - **https://teratermproject.github.io/index-en.html**

2) When you connect the VE2302 Development Kit USB-Type-C JTAG/UART port **J53** to your host Windows PC with the board having been powered on (see Powering On section), the proper drivers, if installed, will give you confirmation of the available COM ports in the Device Manager of Windows as show in the following figure. Review available COM ports prior to power on so the user may identify the newly assigned COM ports for the attached VE2302 Development Kit.

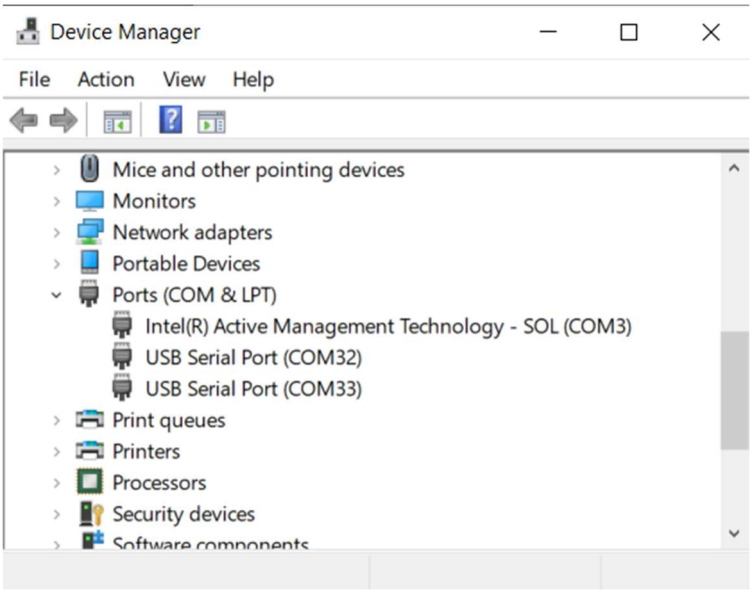

Figure 5 – USB-JTAG/UART COM Port Assignments

3) If the drivers are not installed, then you must manually install the driver for the FT2232H device. Visit the FTDI website and download the appropriate driver for your operating system.

   - **http://www.ftdichip.com/Drivers/VCP.htm**

4) Prior to installing drivers manually, unplug the Type-C cable from the VE2302 Development Kit. After the Kit is unplugged from the PC you can unzip and install the FT2232H driver.

5) At this point you should reboot your PC if you manually installed the FT2232H driver and then proceed to plug in the Type-C cable for the USB JTAG/UART to **J53**.

4.4 Ethernet Connection Setup

Prior to booting the Petalinux Out-of-Box Example Design, we need to ensure that the point-to-point network connection that we are using for this design is setup properly for communication. In this tutorial we happen to be utilizing a Windows laptop. For this to function we need to setup a local LAN with the Windows laptop's ethernet port having an IP Address of **192.168.2.10.**

To-do this in Windows 11 you must go to the **Network & Internet Settings**. From there a user can select the appropriate ethernet port adapter and click on the **Edit** button to get to **Edit IP Settings**.

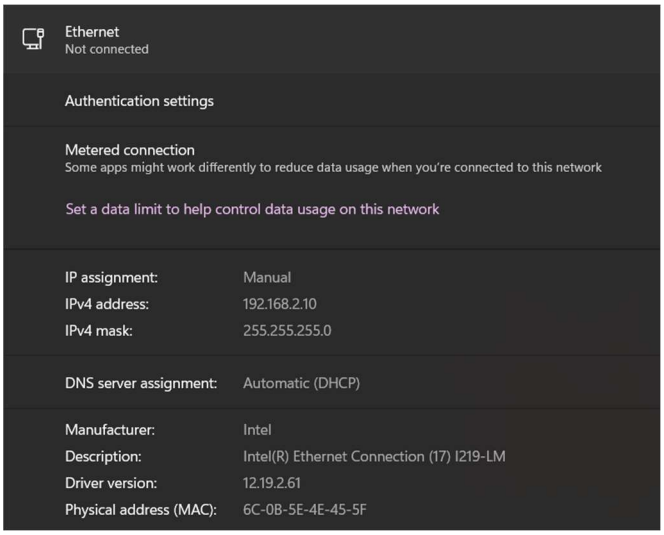

Figure 6 – Ethernet Adapter IP Settings

Set the appropriate IP address in the **Edit IP Settings** area. Here we can set the TCP/IPv4 properties by typing in the IP address we are targeting which is **192.168.2.10**.

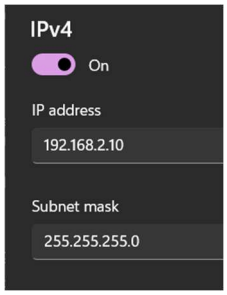

Figure 7 – TCP/IPv4 Properties

4.5 Powering On – Boot Petalinux

The VE2302 Development Kit provides a 12V / 5A power supply to power the kit. The power supply plugs into the 6-pin power receptacle (**J39**) to supply the +12V power source to VE2302 Development Kit.

NOTE: On Revision 1 Versal AI Edge Carrier Card, plugging in the 12V power supply will cause the fan to enable as the FAN PWM signal is not controlled yet and 12V is provided to the FAN HEADER. This is resolved in future revisions of the board.

To power up the VE2302 Development Kit a user needs to toggle the power switch, **SW8**.

When the board is powered on properly, you will see a GREEN power good LED, **D42**, a BLUE DONE LED, **D36**, an ORANGE JTAG Enabled LED, **D34**, and several RED USER LEDs illuminate on the Versal AI Edge Carrier Card.

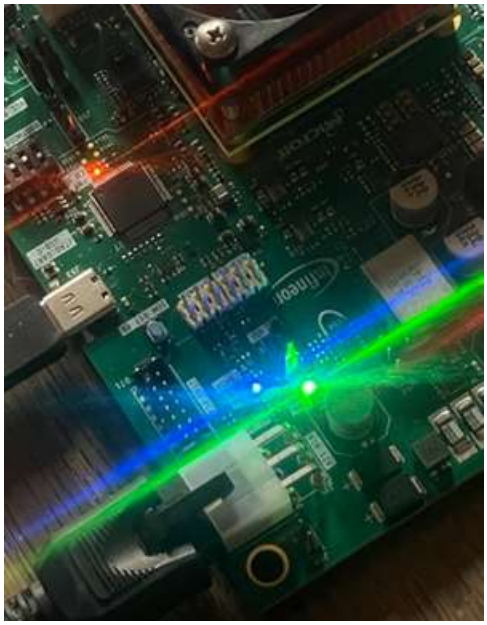

Figure 8 – JTAG/UART, DONE, and POWER GOOD LEDs

With everything setup properly, the board will start to boot and complete with terminal appearing as follows:

```
[  OK  ] Finished Record Runlevel Change in UTMP.
[   19.273649] macb ff0c0000.ethernet end0: Link is Up - 1Gbps/Full - flow control tx
[   19.318014] EXT4-fs (mmcblk0p2): mounted filesystem acbb9e51-3e44-480a-840a-1a4d4a4e6bf5 r/w with ordered data mode. Quota mode: none.
[  OK  ] Mounted /run/media/user-mmcblk0p2.

********************************************************************************
The PetaLinux source code and images provided/generated are for demonstration purposes only.
Please refer to https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/2741928025/Moving+from+PetaLinux+to+Production+Deployment
for more details.
********************************************************************************
PetaLinux 2024.2+release-S11061705 ve2302dkoob ttyAMA0

ve2302dkoob login: root (automatic login)

[   24.175783] audit: type=1006 audit(1709054781.068:2): pid=725 uid=0 old-auid=4294967295 auid=0 tty=(none) old-ses=4294967295 ses=1 res=1
[   24.188131] audit: type=1300 audit(1709054781.068:2): arch=c00000b7 syscall=64 success=yes exit=1 a0=8 a1=ffffc3ab4c30 a2=1 a3=1 items=0
suid=0 egid=0 sgid=0 fsgid=0 tty=(none) ses=1 comm="(systemd)" exe="/usr/lib/systemd/systemd-executor" key=(null)
[   24.214628] audit: type=1327 audit(1709054781.068:2): proctitle="(systemd)"
root@ve2302dkoob:~#
```

Figure 9 – Out-of-Box Design Boot

4.6 Using Out-of-Box Example Design

At this point, it is time to utilize the Out-of-Box Example Design. Open your favourite web browser and point the browser to the IP address of the VE2302 Development Kit: **192.168.2.2**

http://192.168.2.2

If everything has been successful to this point, the user will see the following web page in the web browser:

NOTE: It has been noted in Windows 11 that you may need to turn OFF the laptops WIFI for the RJ45 Ethernet Port / P2P Lan to function.

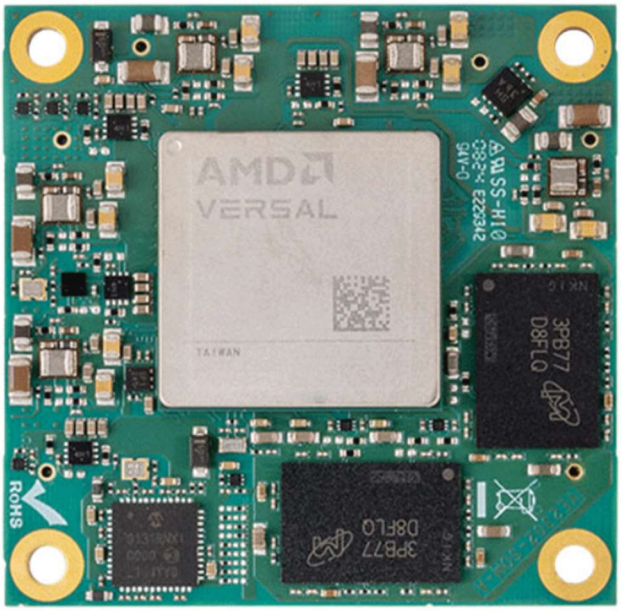

Figure 10 – Out-of-Box Design Web Page

In the webpage, the user will be able to navigate to the VE2302 SOM Product Brief and the VE2302 DEVELOPMENT KIT Product Brief.

From the terminal, the user will be able to perform Linux accesses to GPIO, mount and review memory including storage locations such as the SD card, eMMC, and a flash drive connected to the USB2.0 Type A interface.

Here is an example of USB Flash Drive being enumerated in Linux when plugged into the USB2.0 Type A port, **J6** on the Versal AI Edge Carrier Card:

```
ADDR and VAL may be specified as hex values
root@ve2302dkoob:/# [  572.259321] usb 1-1: new high-speed USB device number 2 using xhci-hcd
[  572.420943] usb 1-1: New USB device found, idVendor=0951, idProduct=1666, bcdDevice= 0.01
[  572.429174] usb 1-1: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[  572.436346] usb 1-1: Product: DataTraveler 3.0
[  572.440822] usb 1-1: Manufacturer: Kingston
[  572.445035] usb 1-1: SerialNumber: 60A44C413C9CF1C0B98F00A1
[  572.456575] usb-storage 1-1:1.0: USB Mass Storage device detected
[  572.464655] scsi host0: usb-storage 1-1:1.0
[  573.480334] scsi 0:0:0:0: Direct-Access     Kingston DataTraveler 3.0      PQ: 0 ANSI: 6
[  573.490861] sd 0:0:0:0: [sda] 15109516 512-byte logical blocks: (7.74 GB/7.20 GiB)
[  573.499437] sd 0:0:0:0: [sda] Write Protect is off
[  573.504585] sd 0:0:0:0: [sda] Write cache: disabled, read cache: enabled, doesn't support DPO or FUA
[  573.519705]  sda:
[  573.521816] sd 0:0:0:0: [sda] Attached SCSI removable disk
[  576.065081]  sda:
```

Figure 11 – USB Flash Drive Enumeration

Here is an example of mounting the USB Flash Drive in Linux and looking at the contents of the USB Flash Drive:

```
root@ve2302dkoob:/# mount /dev/sda /media
root@ve2302dkoob:/# cd /media
root@ve2302dkoob:/media# ls -all
total 125976
drwxrwx---   11 root     disk          4096 Jan  1  1970 .
drwxr-xr-x   16 root     root           400 Mar  9  2018 ..
drwxrwx---    5 root     disk          4096 Nov 26  2025 .Trash-1000
drwxrwx---    2 root     disk          4096 Dec  9  2025 25gbe_demo
drwxrwx---    2 root     disk          4096 Oct  9  2025 FOUND.000
drwxrwx---    2 root     disk          4096 Sep 20  2025 System Volume Information
drwxrwx---    2 root     disk          4096 Nov  4  2025 TRIA_VE2302_SOM_2024_2_FAT_RELEASE_V1P2
-rwxrwx---    1 root     disk     128955568 Nov  4  2025 TRIA_VE2302_SOM_2024_2_FAT_RELEASE_V1P2.zip
drwxrwx---    2 root     disk          4096 Dec  9  2025 oob_demo
drwxrwx---    2 root     disk          4096 Oct  6  2025 oob_prod
drwxrwx---    2 root     disk          4096 Nov  3  2025 oob_prot
drwxrwx---    2 root     disk          4096 Nov  7  2025 ve2302_fat_certification
root@ve2302dkoob:/media#
```

Figure 12 – USB Flash Drive Mounting and Contents

Similarly, you can mount an SD card that is inserted into the SD card slot, **J5**. Here is an example of an appropriate SD card being enumerated in Linux:

```
root@ve2302dkoob:~# [   56.286336] mmc1: new high speed SDHC card at address 5048
[   56.297342] mmcblk1: mmc1:5048 SD32G 29.0 GiB
[   56.304428]  mmcblk1: p1

root@ve2302dkoob:~#
```

Figure 13 – SD Card Enumeration

Here is an example of mounting the SD Card in Linux and looking at the contents of the SD Card:

```
root@ve2302dkoob:~# mount /dev/mmcblk1p1 /mnt
root@ve2302dkoob:~# cd /mnt
root@ve2302dkoob:/mnt# ls -all
total 248480
drwxrwx---    6 root     disk         32768 Jan  1  1970 .
drwxr-xr-x   16 root     root           400 Mar  9  2018 ..
-rwxrwx---    1 root     disk       2263184 Jan  8  2026 BOOT.BIN
-rwxrwx---    1 root     disk      31971840 Jan  8  2026 Image
-rwxrwx---    1 root     disk      12660399 Jan  8  2026 Image.gz
drwxrwx---    2 root     disk         32768 Oct 13  2025 System Volume Information
drwxrwx---    2 root     disk         32768 Oct 13  2025 VE2302 IOCC V1P1
drwxrwx---    2 root     disk         32768 Oct 17  2025 VE2302 SOM V1P1
drwxrwx---    2 root     disk         32768 Oct 31  2025 VE2302 SOM V1P2
-rwxrwx---    1 root     disk          3835 Jan  8  2026 boot.scr
-rwxrwx---    1 root     disk      44051783 Jan  8  2026 image.ub
-rwxrwx---    1 root     disk     100479488 Jan  8  2026 rootfs.cpio
-rwxrwx---    1 root     disk      31350336 Jan  8  2026 rootfs.cpio.gz
-rwxrwx---    1 root     disk      31350400 Jan  8  2026 rootfs.cpio.gz.u-boot
root@ve2302dkoob:/mnt#
```

Figure 14 – SD Card Mounting and Contents

The end user can manipulate the GPIO attached to the RED USER LEDs, Push Buttons, and SWITCHES via commands in Linux. Here is a list of the GPIO mapped in the Vivado HW Design:

```
root@ve2302dkoob:/# cd /sys/class/gpio
root@ve2302dkoob:/sys/class/gpio# ls
export       gpiochip512  gpiochip516  gpiochip518  gpiochip520  gpiochip521  gpiochip579  gpiochip695  gpiochip703  unexport
root@ve2302dkoob:/sys/class/gpio# ls -all
total 0
drwxr-xr-x    2 root     root            0 Feb 27 17:26 .
drwxr-xr-x   71 root     root            0 Feb 27 17:26 ..
--w-------    1 root     root         4096 Feb 27 17:31 export
lrwxrwxrwx    1 root     root            0 Feb 27 17:26 gpiochip512 -> ../../devices/platform/pl-bus/20100000000.gpio/gpio/gpiochip512
lrwxrwxrwx    1 root     root            0 Feb 27 17:26 gpiochip516 -> ../../devices/platform/pl-bus/20180000000.gpio/gpio/gpiochip516
lrwxrwxrwx    1 root     root            0 Feb 27 17:26 gpiochip518 -> ../../devices/platform/pl-bus/20200000000.gpio/gpio/gpiochip518
lrwxrwxrwx    1 root     root            0 Feb 27 17:26 gpiochip520 -> ../../devices/platform/pl-bus/20280000000.gpio/gpio/gpiochip520
lrwxrwxrwx    1 root     root            0 Feb 27 17:26 gpiochip521 -> ../../devices/platform/axi/ff0b0000.gpio/gpio/gpiochip521
lrwxrwxrwx    1 root     root            0 Feb 27 17:26 gpiochip579 -> ../../devices/platform/axi/f1020000.gpio/gpio/gpiochip579
lrwxrwxrwx    1 root     root            0 Feb 27 17:26 gpiochip695 -> ../../devices/platform/axi/ff020000.i2c/i2c-0/0-0044/gpio/gpiochip695
lrwxrwxrwx    1 root     root            0 Feb 27 17:26 gpiochip703 -> ../../devices/platform/axi/f1000000.i2c/i2c-2/2-0043/gpio/gpiochip703
--w-------    1 root     root         4096 Feb 27 17:31 unexport
root@ve2302dkoob:/sys/class/gpio#
```

Figure 15 – GPIO Address Mapping

Here are the mappings of the AXI GPIO blocks according to the addresses assigned:

XPIO DIPSWTICHES – AXI_GPIO_0 – Address 0x20100000000 – GPIOCHIP512 – INPUTS – QTY:4

XPIO LEDS – AXI_GPIO_1 – Address 0x20180000000 – GPIOCHIP516 – OUTPUTS – QTY:2

XPIO PUSH BUTTONS – AXI_GPIO_2 – Address 0x20200000000 – GPIOCHIP518 – INPUTS – QTY:4

MIO LEDS – OFFSETs 47 and 48 from GPIOCHIP579 – OUTPUTS – QTY:2

MIO PUSH BUTTONS – OFFSETS 37 and 46 from GPIOCHIP579 – INPUTS – QTY:2

I2C GPIO LEDS – OFFSET 6 and 7 from I2C GPIO EXPANDER CHIP – I2C ADDRESS 0x43 – QTY:2

An example of toggling LEDs ON and OFF that are attached to PL GPIO pins is executed by implementing the following command in Linux:

```
root@ve2302dkoob:/sys/class/gpio# gpio-demo -g 516 -o 0
root@ve2302dkoob:/sys/class/gpio# gpio-demo -g 516 -o 1
root@ve2302dkoob:/sys/class/gpio# gpio-demo -g 516 -o 3
root@ve2302dkoob:/sys/class/gpio#
```

Figure 16 – Toggling XPIO LEDs ON/OFF

An example of Linux reading the values set on the 4-position dipswitch, **SW5,** that are attached to PL GPIO pins is executed by implementing the following command in Linux. This command shows various reads after toggling the 4-position dipswitch to different settings like ALL-ON, ALL-OFF, and alternating.

```
root@ve2302dkoob:/sys/class/gpio#
root@ve2302dkoob:/sys/class/gpio#
root@ve2302dkoob:/sys/class/gpio# gpio-demo -g 512 -i
0x0000000F
root@ve2302dkoob:/sys/class/gpio# gpio-demo -g 512 -i
0x00000000
root@ve2302dkoob:/sys/class/gpio# gpio-demo -g 512 -i
0x00000005
root@ve2302dkoob:/sys/class/gpio#
```

Figure 17 – Reading XPIO DIPSWITCH Values

An example of Linux reading the values set by engaging the 2 XPIO push buttons, **SW3 and SW4,** that are attached to PL GPIO pins is executed by implementing the following command in Linux. This command shows various reads after not engaging the push buttons and after engaging (holding down) one or both XPIO push buttons.

```
root@ve2302dkoob:/sys/class/gpio#
root@ve2302dkoob:/sys/class/gpio# gpio-demo -g 518 -i
0x00000000
root@ve2302dkoob:/sys/class/gpio# gpio-demo -g 518 -i
0x00000001
root@ve2302dkoob:/sys/class/gpio# gpio-demo -g 518 -i
0x00000002
root@ve2302dkoob:/sys/class/gpio# gpio-demo -g 518 -i
0x00000003
root@ve2302dkoob:/sys/class/gpio#
```

Figure 18 – Reading XPIO PUSH BUTTON Values

An example of toggling LEDs ON and OFF that are attached to PMC MIO pins is executed by implementing the following command in Linux. This targets the MIO47 pin that is offset from the MIO Base GPIOCIHP579:

```
root@ve2302dkoob:/sys/class/gpio#
root@ve2302dkoob:/sys/class/gpio# echo 626 > /sys/class/gpio/export
root@ve2302dkoob:/sys/class/gpio# echo out > /sys/class/gpio/gpio626/direction
root@ve2302dkoob:/sys/class/gpio# echo 1 > /sys/class/gpio/gpio626/value
root@ve2302dkoob:/sys/class/gpio# echo 0 > /sys/class/gpio/gpio626/value
root@ve2302dkoob:/sys/class/gpio# echo 1 > /sys/class/gpio/gpio626/value
root@ve2302dkoob:/sys/class/gpio# echo 0 > /sys/class/gpio/gpio626/value
root@ve2302dkoob:/sys/class/gpio#
root@ve2302dkoob:/sys/class/gpio#
```

Figure 19 – Toggling PMC MIO LED at MIO47

An example of toggling LEDs ON and OFF that are attached to PMC MIO pins is executed by implementing the following command in Linux. This targets the MIO48 pin that is offset from the MIO Base GPIOCIHP579:

```
root@ve2302dkoob:/sys/class/gpio# echo 627 > /sys/class/gpio/export
root@ve2302dkoob:/sys/class/gpio# echo out > /sys/class/gpio/gpio627/direction
root@ve2302dkoob:/sys/class/gpio# echo 1 > /sys/class/gpio/gpio627/value
root@ve2302dkoob:/sys/class/gpio# echo 0 > /sys/class/gpio/gpio627/value
root@ve2302dkoob:/sys/class/gpio# echo 1 > /sys/class/gpio/gpio627/value
root@ve2302dkoob:/sys/class/gpio# echo 0 > /sys/class/gpio/gpio627/value
```

Figure 20 – Toggling PMC MIO LED at MIO48

An example of Linux reading the values set by engaging the 2 PMC MIO push buttons, **SW1 and SW2,** that are attached to PMC MIO pins is executed by implementing the following command in Linux. This command shows various reads after not engaging the push buttons and after engaging (holding down) PMC MIO push button.

```
root@ve2302dkoob:/sys/class/gpio# echo 616 > /sys/class/gpio/export
root@ve2302dkoob:/sys/class/gpio# echo 625 > /sys/class/gpio/export
root@ve2302dkoob:/sys/class/gpio# echo in > /sys/class/gpio/gpio616/direction
root@ve2302dkoob:/sys/class/gpio# echo in > /sys/class/gpio/gpio625/direction
root@ve2302dkoob:/sys/class/gpio# cat ./gpio616/value
0
root@ve2302dkoob:/sys/class/gpio# cat ./gpio616/value
1
root@ve2302dkoob:/sys/class/gpio# cat ./gpio625/value
0
root@ve2302dkoob:/sys/class/gpio# cat ./gpio625/value
1
root@ve2302dkoob:/sys/class/gpio#
```

Figure 21 – Reading PMC MIO PUSH BUTTON Values

An example of toggling LEDs ON and OFF that are attached to I2C GPIO expander chip, **U13**, on the VE2302 SOM. Toggling these LEDs, **D3 and D7**, is executed by implementing the following command in Linux. This targets the I2C GPIO expander pins 6 and 7 that contains RED USER LEDs on the Versal AI Edge Carrier Card. The I2C address of the I2C GPIO expander chip is **0x43**, the I2C bus that the I2C GPIO expander chip resides on is **0x2**, register address **0x3** is the GPIO direction setting for the I2C GPIO expander chip, register address **0x5** is the GPIO value setting for the I2C GPIO expander chip.

```
root@ve2302dkoob:/sbin#
root@ve2302dkoob:/sbin# i2cset -f 2 0x43 0x3 0xc0
WARNING! This program can confuse your I2C bus, cause data loss and worse!
I will write to device file /dev/i2c-2, chip address 0x43,
data address 0x03, data 0xc0, mode byte.
Continue? [Y/n] y
root@ve2302dkoob:/sbin# i2cset -f 2 0x43 0x5 0xc0
WARNING! This program can confuse your I2C bus, cause data loss and worse!
I will write to device file /dev/i2c-2, chip address 0x43,
data address 0x05, data 0xc0, mode byte.
Continue? [Y/n] y
root@ve2302dkoob:/sbin# i2cset -f 2 0x43 0x5 0x00
WARNING! This program can confuse your I2C bus, cause data loss and worse!
I will write to device file /dev/i2c-2, chip address 0x43,
data address 0x05, data 0x00, mode byte.
Continue? [Y/n] y
root@ve2302dkoob:/sbin#
```

Figure 22 – Toggling I2C GPIO LED Values

4.7 Getting Started Complete

That completes this tutorial. The user may now power down the VE2302 Development Kit by toggling power switch **SW8** to the OFF position.

5 Getting Help and Support

If additional support is required, TRIA Technologies has many avenues to search depending on your needs.

For general questions regarding VE2302 Development Kit, please visit our website at **http://avnet.me/ve2302-dk**. Here you can find any available documentation, technical specifications, videos and tutorials, reference designs and other support.

Detailed questions regarding VE2302 Development Kit hardware design, software application development, using AMD tools, training and other topics can be posted on the VE2302 Development Kit Support Forum at **http://avnet.me/ve2302-dk-forum**. Avnet's technical support team monitors the forum during normal business hours in North America.

Those interested in available customization options on VE2302 Development Kit can send inquiries to **customize@avnet.com**.