

VCS³ Stack

Getting started guide

Table of Contents

Kit contents	2
Hardware Setup	3
The VCS ³ -RP2040	4
About the VCS ³ -RP2040	4
Using the VCS3-RP2040 as a XVC JTAG pod.	5
Setting up the software under Linux	5
Under Windows	6
Using a XVC in Vivado.	7
Using the VCS ³ -RP2040 as a UART connection.....	10
Under LINUX.....	10
Under Windows	10
Using the built in Demo	11
About the VCS3.....	12
Description	12
Parts of the VCS ³	14
Boot Options.....	15

Version	Notes	Initials	Date
0.1	Initial Version based on VCS3 3D kit.	CH	12 th Oct. 2026

Kit contents

The VCS3 stack is a compact single unit with multiple I/Os already connected.

It is comprised of the unit itself and a 5V USB-C power supply (UK customers only) – Not pictured.

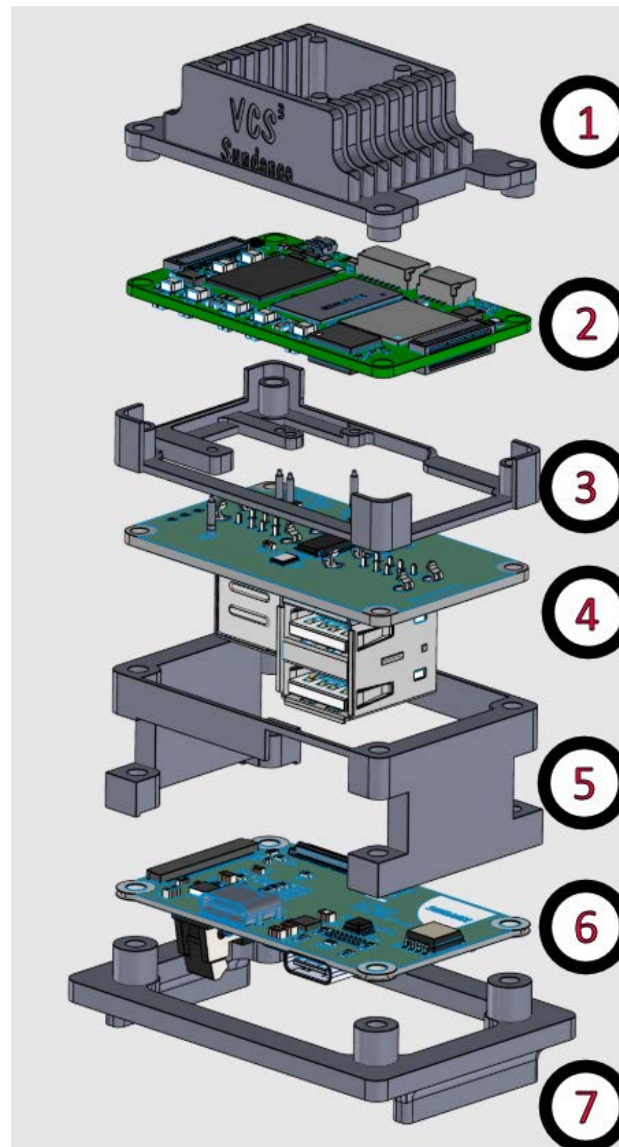


Figure 1 Exploded Stack

1. 3D printed fan housing
2. VCS3 Core Module
3. 3D printed support
4. VCS3-USB Module
5. 3D printed support
6. VCS3-RP2040-50x30 Module
7. 3D printed foot

VCS3-RP2040-50x30

The VCS3-RP2040-50x30 is a Pi RP2040 powered converter card for DSI MIPI interface adaption, UART communication and XVC JTAG.

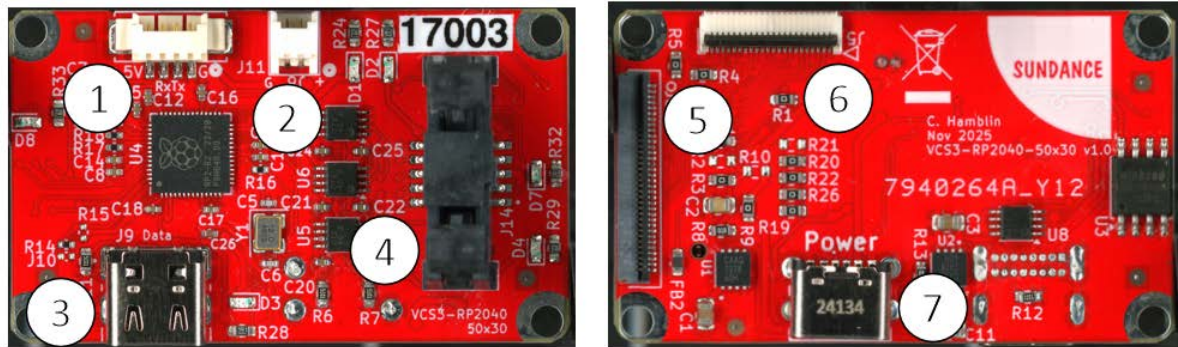


Figure 2 VCS3_RP2040-50x30 Module

1. Power and UART connector for VCS3 Core module
2. Fan power
3. USB-C connector for UART and AMD XVC JTAG communication
4. XVC JTAG connector
5. DSI MIPI conversion (out to screen)
6. DSI MIPI conversion (in from VCS3 Core)
7. USB-C connector for powering VCS3 Core module

VCS3-USB Module

The VCS3-USB module utilises the pads on the bottom of the VCS3 to connect a USB2 hub via pogo pins. Via the USB hub chip on board, it expands the VCS3 to be able to have 4x USB 2 connections at one time.

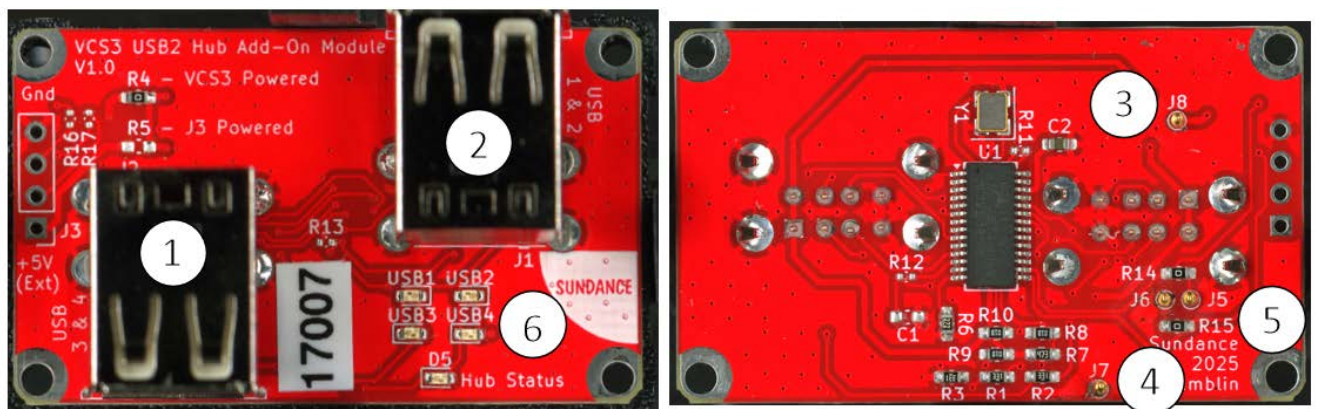


Figure 3 VCS3-USB Module

1. Dual USB-A connector
2. Dual USB-A connector
3. PoGo pin for Power
4. PoGo Pin for Gnd
5. PoGo pins for USB data
6. LED indicators

The VCS³-RP2040-50x30

About the VCS³-RP2040-50x30

The I/O card the VCS³-RP2040 is used for XVC JTAG as well as UART communication to the VCS³ module.

It will come programmed with the required code to run the UART and XVC JTAG. If required, a link to our fork of the master GitHub repository can be provided for re-programming.

It is firmware and software is based on the GitHub project xvc-pico, but has been forked. (<https://github.com/SundanceMultiprocessorTechnology/xvc-vcs3-rp2040>).

The VCS³-RP2040-50x30 converts the 22 pin MIPI on from the VCS³ to the 34 pin connector for use with a 10.1" screen (Not supplied).

Using the VCS3-RP2040-50x30 as a XVC JTAG pod.

Setting up the software under Linux

(Taken from <https://github.com/kholia/xvc-pico>)

Building pico-xvc (for Linux users)

Install dependencies:

```
sudo apt install cmake gcc-arm-none-eabi libnewlib-arm-none-eabi \
    libstdc++-arm-none-eabi-newlib git libusb-1.0-0-dev build-essential \
    make g++ gcc
mkdir ~/repos
cd ~/repos
```

```
git clone https://github.com/raspberrypi/pico-sdk.git
cd pico-sdk; git submodule update --init
```

```
cd ~/repos
```

```
git clone https://github.com/SundanceMultiprocessorTechnology/xvc-vcs3-
rp2040.git
```

Build the host-side daemon:

```
cd ~/repos/xvc-vcs3-rp2040/daemon
cmake .
make
```

You only need to this once.

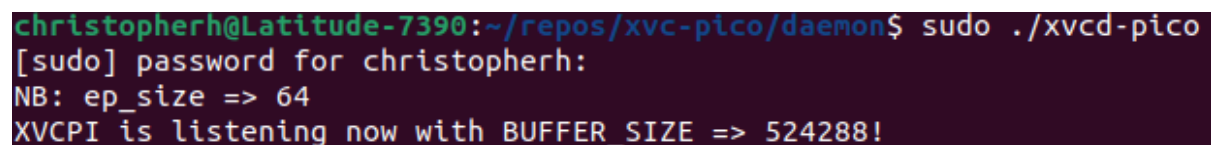
Now, go to the right directory and run the daemon to enable the host side XVC software:

```
cd ~/repos/ xvc-vcs3-rp2040/daemon
```

Connect the VCS³-RP2040 to your computer if not already, and then run

```
sudo ./xvcd-pico
```

After entering your password you should see the following:



```
christopherh@Latitude-7390:~/repos/xvc-pico/daemon$ sudo ./xvcd-pico
[sudo] password for christopherh:
NB: ep_size => 64
XVCPI is listening now with BUFFER_SIZE => 524288!
```

Figure 4 output of the command under LINUX

Leave this running and then launch the AMD tool you require. We are going to show you the procedure to use vivado hardware manager.

Under Windows

Grab **xvcd-pico.exe** from the builds folder of this repository itself.

(<https://github.com/SundanceMultiprocessorTechnology/xvc-vcs3-rp2040/tree/ng/builds>)

You need to install the libusbK driver with Zadig (<https://zadig.akeo.ie/>).

OR

You can install the WinLibUSB driver from USB Drive Tool Application

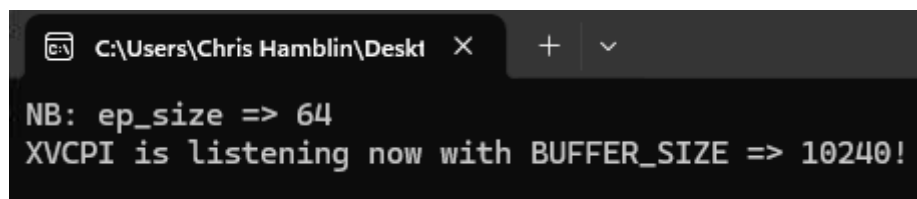
(<https://visualgdb.com/UsbDriverTool/>).

Credit goes to <https://github.com/benitoss/> for these instructions.

Usage

On the host computer with the Raspberry Pi Pico connected, run the XVC Daemon Server (**xvcd-pico.exe**) from where you downloaded it.

You should see a box like this:

A screenshot of a Windows command prompt window. The title bar shows the path 'C:\Users\Chris Hamblin\Desktop' and standard window controls. The command prompt displays two lines of text: 'NB: ep_size => 64' and 'XVCPI is listening now with BUFFER_SIZE => 10240!'.

```
C:\Users\Chris Hamblin\Desktop > xvcd-pico.exe
NB: ep_size => 64
XVCPI is listening now with BUFFER_SIZE => 10240!
```

Figure 5 - output of the .exe under windows

Leave this open while you are using the XVC.

Using a XVC in Vivado.

Open Vivado and select “Open Hardware Manager”:

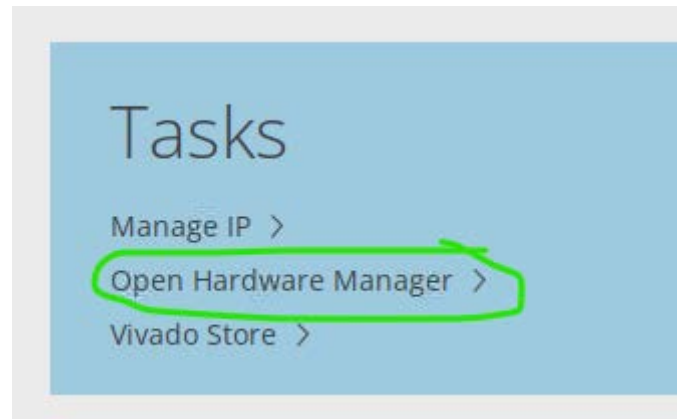


Figure 6 - Vivado Hardware manager

Start the Open New Target wizard: “Tools -> Open new Target”

Press “Next” on the first screen.

On the next screen, ensure that Local Server is selected. Press next.

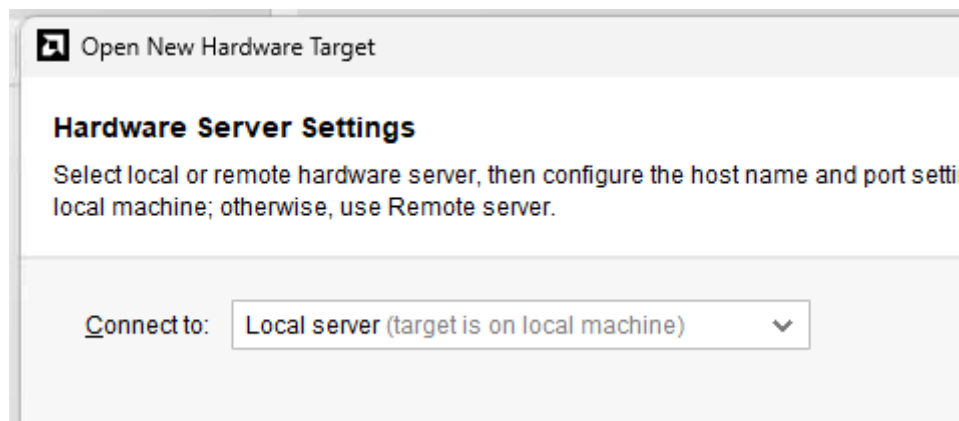


Figure 7 - Open Hardware Wizard Local Server

The next screen will look blank, this is OK.

Press the Add Xilinx Virtual Cable (XVC) button.

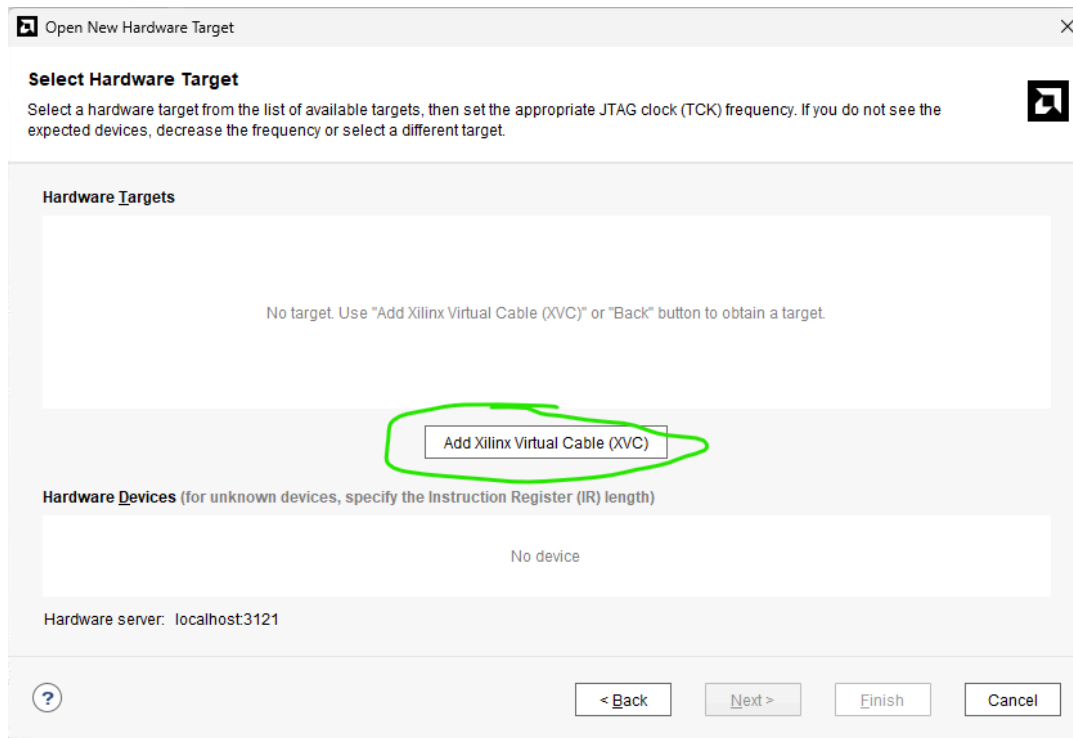


Figure 8 - Add Xilinx Virtual Cable

In the box that pops up, set the host name to “localhost.” Leave the port as 2542.

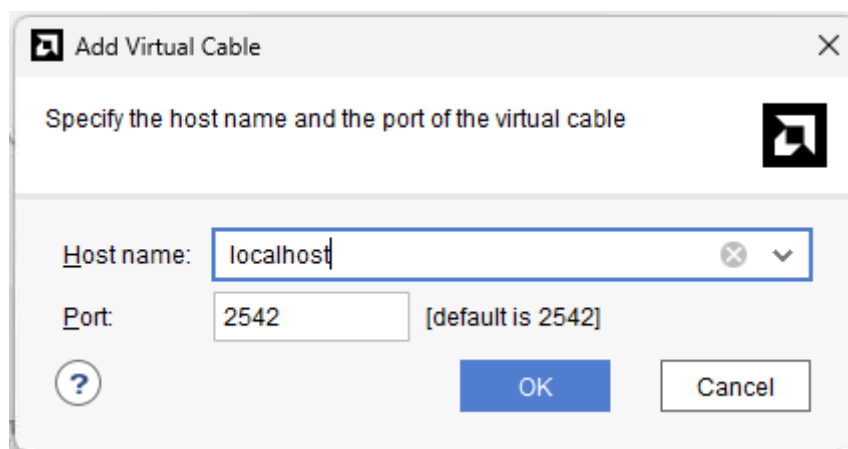


Figure 9 - Virtual Cable Settings

Press OK.

The wizard then connects to the XVC cable and checks what is connected. You will see a clock frequency of 1000000.

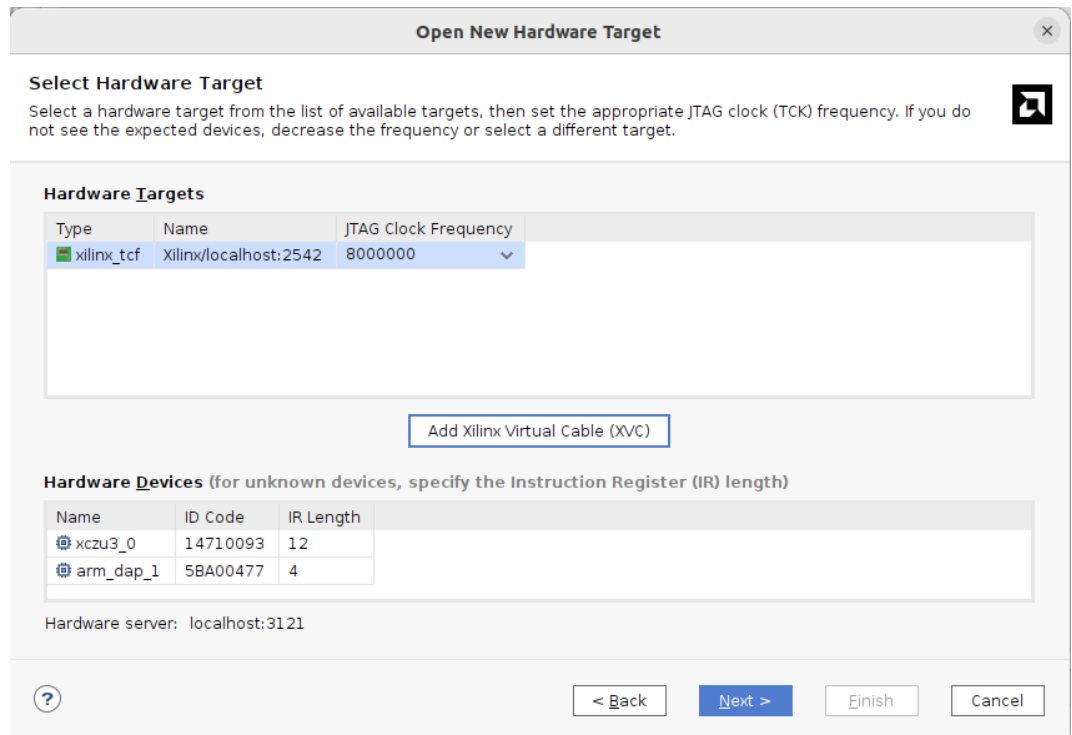


Figure 10 - Virtual Cable: Good Result

It should show like above, where there is a hardware target listed, and two hardware devices show, the xczu3 and the arm core.

Press next, then finish.

It will then connect and refresh the devices. You can now use it as normal.

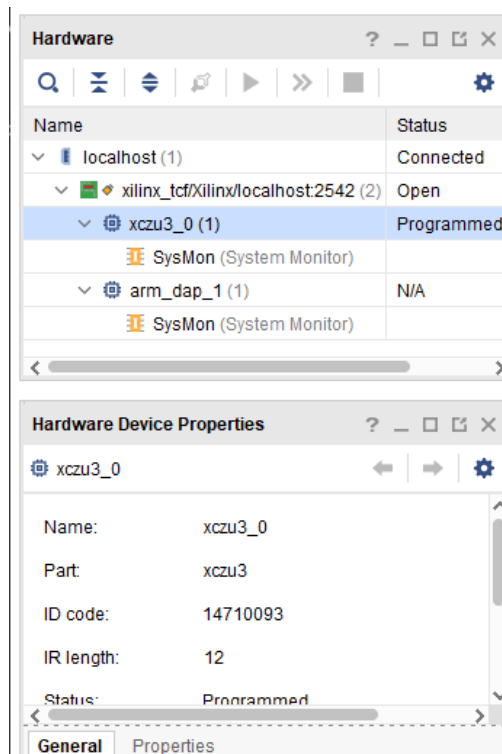


Figure 11 - Vivado Connected Devices

Using the VCS³-RP2040 as a UART connection

Under LINUX

Plug the lefthand USB-C connector into your computer. It runs at less than USB 2 speeds so any USB-A – USB-C cable that supports data is fine, as well as any USB-C – USB-C cable.

Once connected, you should now have a “/dev/ttyACM(n)” device which can be connected by the UART communicator of your choice (screen, minicom, putty etc. etc.).

The connection baud rate is 115200.

If you “lsusb” you should see a “Raspberry Pi Pico” listed.

Under Windows

Plug the lefthand USB-C connector into your computer. It runs at less than USB 2 speeds so any USB-A – USB-C cable that supports data is fine, as well as any USB-C – USB-C cable.

Once connected, you should now have a new comport device listed in the hardware manager which can be connected by the UART communicator of your choice (teraterm, putty etc. etc.).

The connection baud rate is 115200.

Using the VCS3 Stack

There is no built in demo for the VCS3 stack, but there is a version of petelinux installed in the eMMC device, and the VCS3 is set to boot from this by default.

Connect the USB-C for UART & XVC JTAG first, and open the terminal of your choice.

Connect the second USB-C cable and the system will boot and the fan will start spinning.

You will see the boot progress in the UART terminal window.

It is set to auto login, and there is no set username or password.



Figure 12 VCS3-Stack with 4x USB memory sticks connected and detected

About the VCS3

Description

The VCS3 is ‘probably’ the smallest single-board computer based on an AMD Zynq MPSoC.

An ultra-compact, low-power, vision, control, and sensors solution for precision robotics.

The VCS³ is a small single-board computer with an AMD© ZYNQ™ device with integrated ARM CPUs and FPGA fabric. Measuring just 30mm x 50mm, this tiny workhorse can be placed almost anywhere, opening the benefits of FPGAs to many more applications.

The VCS³ utilizes an AMD UltraScale+ MPSoC coupled with high-speed LP-DDR4 memory to produce a highly compact evaluation platform. Together with four digital camera interfaces, a 9-axis IMU, and a CAN-bus interface, this platform is ideally suited for autonomous machines, cameras, or automation.

Device booting can be from SPI ROMs, eMMC flash or for development, JTAG.

Numerous onboard power rails are generated from a single external 5V supply, either from a dedicated connector or via a USB3 Type-C interface.

Several LEDs indicate board functionality, and eight connectors allow access to the various interfaces.

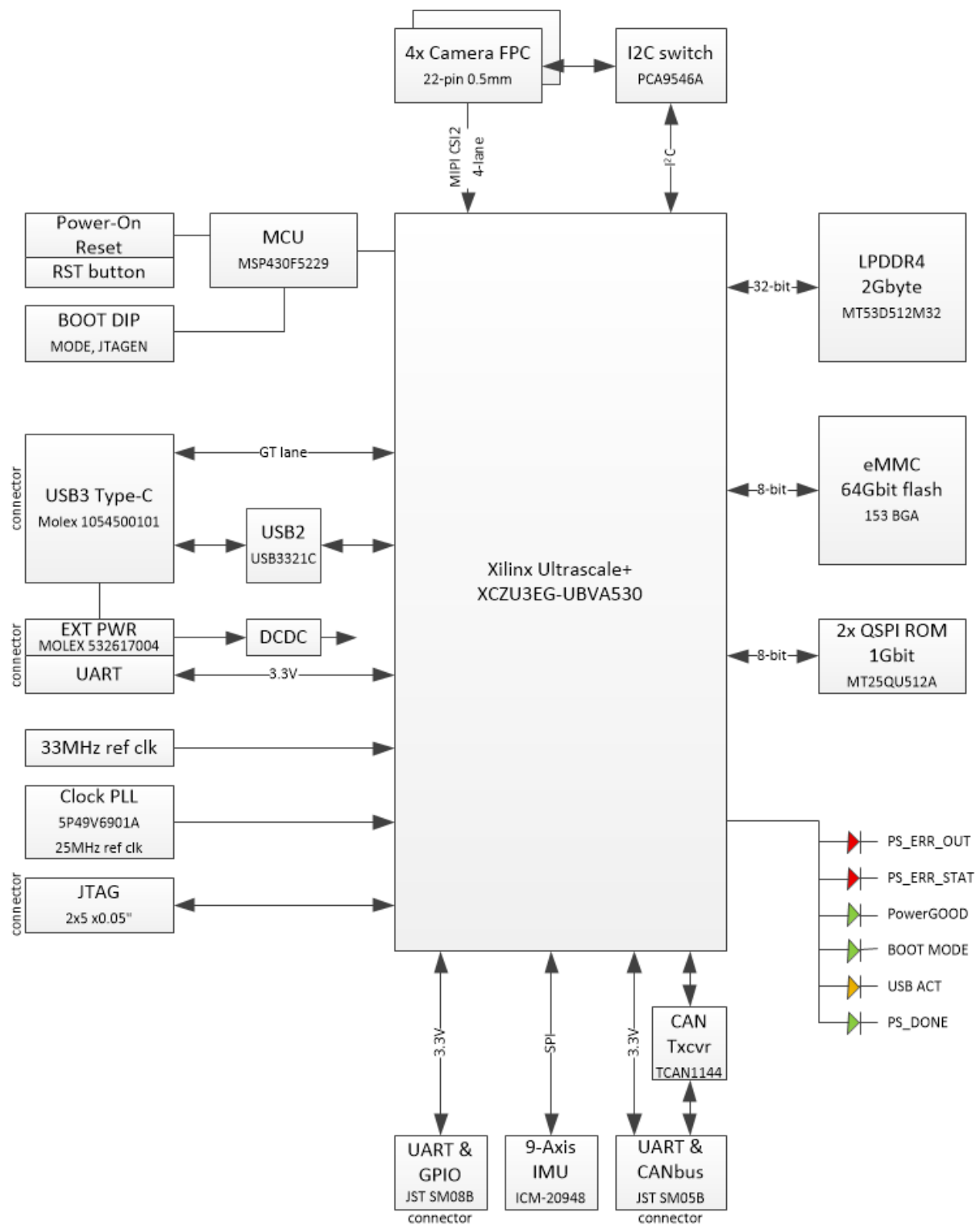


Figure 13 Block Diagram of the VCS³

Parts of the VCS³

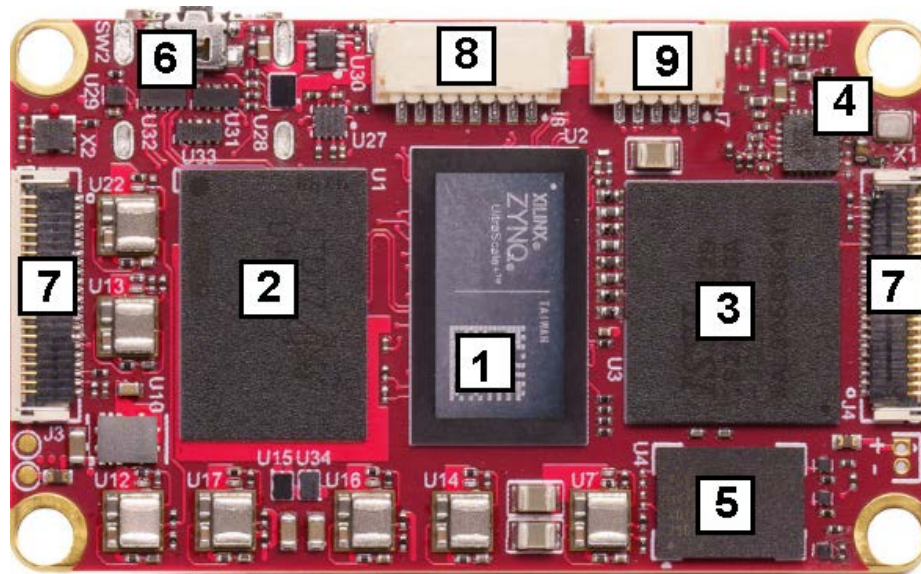


Figure 14 - VCS³ Top

1	AMD Zynq MPSoC	2	LP-DDR4	3	eMMC	4	9-axis IMU	5	SPI ROM
6	Reset Switch	7	MIPI Camera Connectors	8	GPIOs	9	CAN BUS and UART	10	JTAG header
11	BOOT Mode Switch	12	CAN BUS Interface	13	Power and UART	14	Microcontroller	15	USB3 Type C

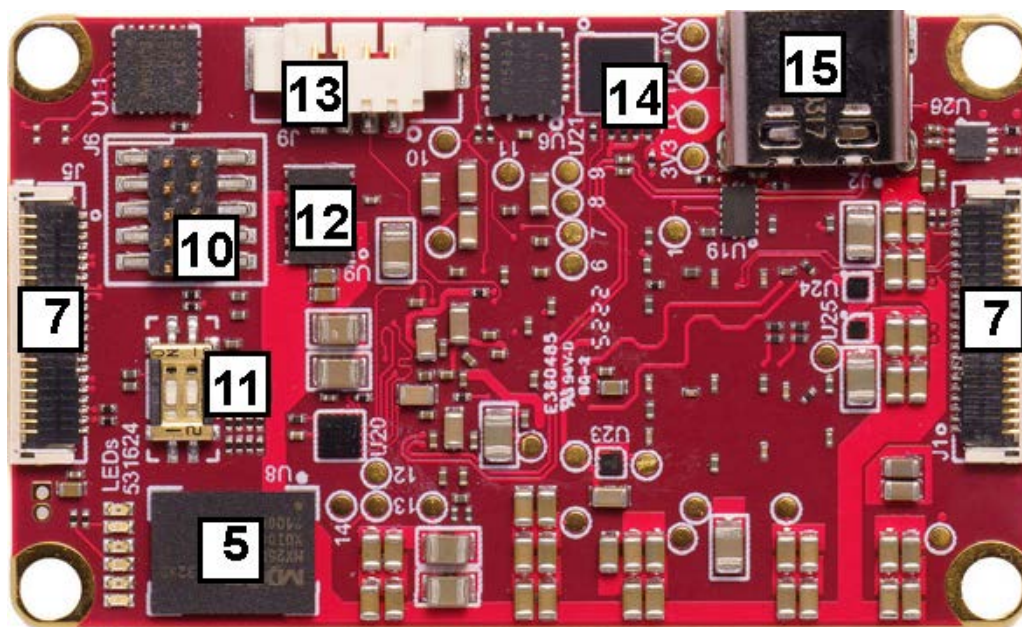


Figure 15 - VCS³ Bottom

Boot Options

There are three boot options at present for the VCS³. These are controlled from SW1 (Item 11 above).

Switch 1	Switch 2	
OFF	OFF	JTAG
ON	OFF	eMMC
OFF	ON	QSPI ROM
ON	ON	