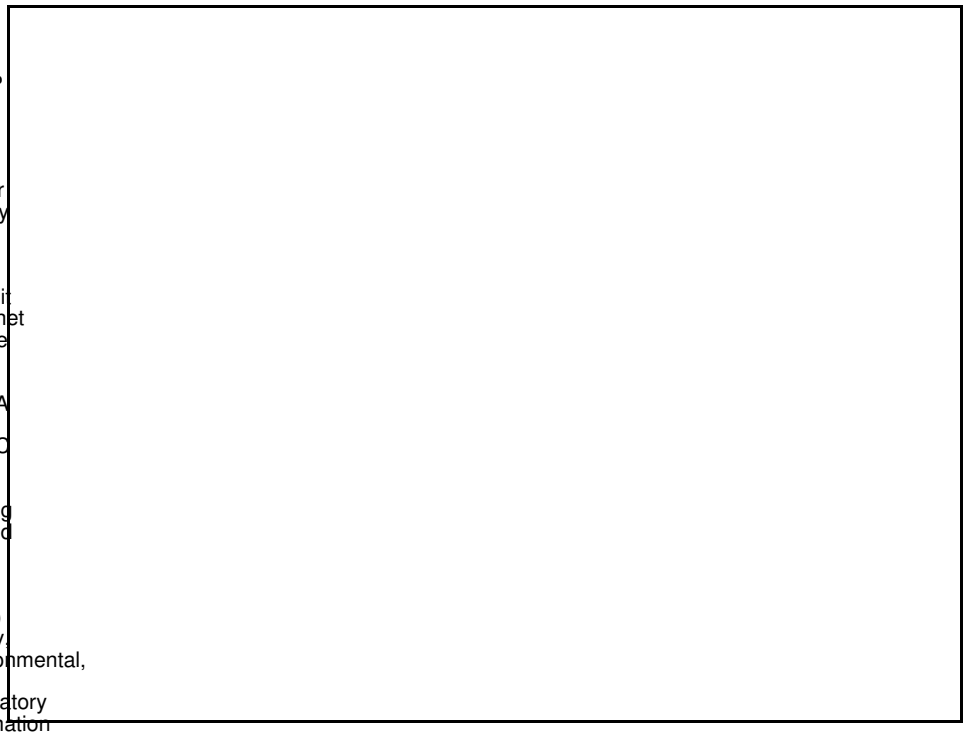
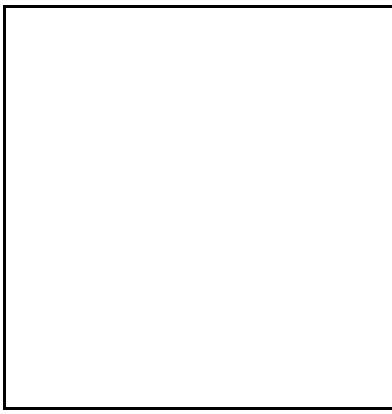
X440 Getting Started Guide

Contents

- 1 Kit Contents
  - 1.1 X4x0
- 2 USRP X440 Design Considerations
- 3 You Will Need
- 4 Proper Care and Handling
- 5 Install and Setup the Software Tools on Your Host Computer
- 6 Assembling the X4x0
- 7 The STM32 Microcontroller
  - 7.1 Updating the SCU
- 8 eMMC Storage
- 9 USB Access to eMMC
- 10 Flashing the eMMC
- 11 Using a USRP X4x0 from UHD
- 12 Updating Filesystems
- 13 Network Interfaces
- 14 Network Connectivity
  - 14.1 Network Status LEDs
    - 14.1.1 RJ45 LED Behavior
    - 14.1.2 QSFP28 LED Behavior
- 15 Security-related Settings
- 16 Serial Connection
- 17 Connecting to the Microcontroller
- 18 SSH Connection
- 19 Autoboot
- 20 Updating the FPGA
- 21 FPGA Image Flavors
- 22 Device Arguments
- 23 GPS
- 24 Front-Panel Programmable GPIOs
- 25 Subdev Specifications
- 26 Rear Panel Status LEDs
  - 26.1 X4x0 Rear Panel Status LEDs
  - 26.2 Power LED Behavior
  - 26.3 User-configurable LEDs

- NI Ettus USRP X410 or X440
- DC Power Supply (12V, 20A)
- 1 Gigabit Ethernet Cat-5e Cable (3m)
- USB-A to USB-C Cable (1m)
- Getting Started Guide URL (QR Code)
- Safety, Environmental, and Regulatory Information





- https://kb.ettus.com/About_Sampling_Rates_and_Master_Clock_Rates_for_the_USRP_X440

- For Network Mode: A host computer with an available 1 or 10 Gigabit Ethernet interface for sample streaming. In addition to the Ethernet interface used for sampling streaming, your host computer will require a separate 1 Gigabit Ethernet interface for command and control streaming.
- For Stand-Alone Embedded Mode: A host computer with an available 1 Gigabit Ethernet port or a USB 2.0 port to remotely access the embedded Linux operating system running on ARM CPU.

All Ettus Research products are individually tested before shipment. The USRP is guaranteed to be functional at the time it is received by the customer. Improper use or handling of the USRP can cause the device to become non-functional. Take the following precautions to prevent damage to the unit.

- Never allow metal objects to touch the circuit board while powered.
- Always properly terminate the transmit port with an antenna or 50? load.
- Always handle the board with proper anti-static methods.
- Never allow the board to directly or indirectly come into contact with any voltage spikes.
- Never allow any water or condensing moisture to come into contact with the device.
- Always use caution with FPGA, firmware, or software modifications.

☐ X410: Never apply more than +14 dBm continuous <=3GHz, +17 dBm continuous >3GHz, or +20dBm more than 5 minutes >3GHz of power into any RF input.

☐ X440: Never apply more than +13 dBm continuous <=2.5GHz, +17 dBm continuous between 2.5GHz and 3.6 GHz, or +20dBm continuous between 3.6 GHz and 4 GHz of power into any RF input.

☐ X410: Always use at least 30dB attenuation if operating in loopback configuration.

In order to use your Universal Software Radio Peripheral (USRP?), you must have the software tools correctly installed and configured on your host computer. The easiest way to install USRP Hardware Driver (UHD) is by getting a binary installer package for your operating system as described in the UHD manual about Binary Installation. If no binary packages are available for your operating system or you want to modify the sources by yourself, a step-by-step guide is available at the Building and Installing the USRP Open-Source Toolchain (UHD and GNU Radio) on Linux, OS X and Windows Application Notes.

To find the latest release of UHD, see the UHD repository at https://github.com/EttusResearch/uhd.

The USRP X410 requires UHD version 4.1 or later. The USRP X440 requires UHD version 4.5 or later.

When you receive a brand-new device, it is strongly recommended that you download the latest filesystem image from the Ettus Research website update the unit. It is not recommended that you use the filesystem from the factory as-is. Instructions on downloading the latest filesystem image and updating it is listed below.

Note that if you are operating the device in Network Mode, the version of UHD running on the host computer and the USRP X4x0 must match.

Inside the kit you will find the X4x0 and an X4x0 power supply. Plug these in, connect the 1GbE RJ45 interface to your network, and power on the device by pressing the power button.

The STM32 microcontroller (also referred to as the "SCU") controls various low-level features of the X4x0 series motherboard: It controls the power sequencing, reads out fan speeds and some of the temperature sensors. It is connected to the RFSoC via an I2C bus. It is running software based on Chromium EC.

It is possible to log into the STM32 using the serial interface (see Connecting to the Microcontroller). This will allow certain low-level controls, such as remote power cycling should the CPU have become unresponsive for whatever reason.

The writable SCU image file is stored on the filesystem under /lib/firmware/ni/ec-titanium-revX.RW.bin (where X is a revision compatibility number). To update, simply replace the .bin file with the updated version and reboot.

The main non-volatile storage of the USRP is an eMMC:

- USRP X410: 16 GB (Module Revision H or earlier) or 32 GB (Module Revision J and later)
- USRP X440: 16 GB (Module Revision D or earlier) or 32 GB (Module Revision E and later)

This storage can be made accessible as a USB Mass Storage device through the USB-OTG connector on the back panel.

The entire root file system (Linux kernel, libraries) and any user data are stored on the eMMC. It is partitioned into four partitions:

Boot partition (contains the bootloader). This partition usually does not require modification. A data partition, mounted in /data. This is the only partition that is not erased during file system updates. Two identical system partitions (root file systems). These contain the operating system and the home directory (anything mounted under / that is not the data or boot partition). The reason there are two of these is to enable remote updates: An update running on one partition can update the other one without any effect to the currently running system. Note that the system partitions are erased during updates and are thus unsuitable for permanently storing information. Note: It is possible to access the currently inactive root file system by mounting it. After logging into the device using serial console or SSH (see the following two sections), run the following commands:

```
$ mkdir temp

$ mount /dev/mmcblk0p3 temp # This assumes mmcblk0p3 is currently not mounted

$ ls temp # You are now accessing the idle partition:

bin   data  etc   lib         media  proc  sbin  tmp    usr
boot  dev   home  lost+found  mnt    run   sys   uboot  var
```

The device node in the mount command might differ, depending on which partition is currently already mounted.

While Mender should be used for routine filesystem updates (see Updating Filesystems), it is also possible to access the X4x0's internal eMMC from an external host over USB. This allows accessing or modifying the filesystem, as well as the ability to flash the device with an entirely new filesystem.

In order to do so, you'll need an external computer with two USB ports, and two USB cables to connect the computer to your X4x0. The instructions below assume a Linux host.

First, connect to the APU serial console at a baud rate of 115200. Boot the device, and stop the boot sequence by typing noautoboot at the prompt. Then, run the following command in the U-boot command prompt:

```
ums 0 mmc 0
```

This will start the USB mass storage gadget to expose the eMMC as a USB mass storage device. You should see a spinning indicator on the console, which indicates the gadget is active.

Next, connect your external computer to the X4x0's USB to PS port using an OTG cable. Your computer should recognize the X4x0 as a mass storage device, and you should see an entry in your kernel logs (dmesg) that looks like this:

```
usb 3-1: New USB device found, idVendor=3923, idProduct=7a7d, bcdDevice= 2.23
usb 3-1: New USB device strings: Mfr=1, Product=2, SerialNumber=0
usb 3-1: Product: USB download gadget
usb 3-1: Manufacturer: National Instruments
sd 6:0:0:0: [sdc] 30932992 512-byte logical blocks: (15.8 GB/14.8 GiB)
sdc: sdc1 sdc2 sdc3 sdc4
sd 6:0:0:0: [sdc] Attached SCSI removable disk
```

The exact output will depend on your machine, but from this log you can see that the X4x0 was recognized and /dev/sdc is the block device representing the eMMC, with 4 partitions detected (see eMMC Storage for details on the partition layout).

It is now possible to treat the X4x0's eMMC as you would any other USB drive: the individual partitions can be mounted and accessed, or the entire block device can be read/written.

Once you're finished accessing the device over USB, the u-boot gadget may be stopped by hitting Ctrl-C at the APU serial console.

Once the X4x0's eMMC is accessible over USB, it's possible to write the filesystem image and thus change the device's filesystem. You can obtain the latest filesystem image by running:

```
uhd_images_downloader -t sdimg -t x4xx
```

The output of this command will indicate where the downloaded images were put, or specify a custom location using using the `-i INSTALL_LOCATION` argument.

There are 2 ways to write the image to the X4x0's eMMC: using `dd` and `bmaptool`. Run one of the following commands, replacing `/dev/sdX` with the block device of the X4x0's eMMC (found in the device's kernel log or by running `lsblk`). Take care to use the correct block device or else you might overwrite the wrong drive!

```
sudo dd if=/path/to/usrp_x4xx_fs.sdimg of=/dev/sdX bs=1M
```

```
sudo bmaptool copy --bmap /path/to/usrp_x4xx_fs.sdimg.bmap /path/to/usrp_x4xx_fs.sdimg /dev/sdX
```

The former is generally preferred as it will always work, even if it slower than the latter.

Like any other USRP, all X4x0 USRPs are controlled by the UHD software. To integrate a USRP X4x0 into your C++ application, you would generate a UHD device in the same way you would for any other USRP:

```
auto usrp = uhd::usrp::multi_usrp::make("type=x4xx");
```

For a list of which arguments can be passed into make(), see Section Device Arguments.

Mender is a third-party software that enables remote updating of the root file system without physically accessing the device (see also the Mender website). Mender can be executed locally on the device, or a Mender server can be set up which can be used to remotely update an arbitrary number of USRP devices. Mender servers can be self-hosted, or hosted by Mender (see mender.io for pricing and availability).

When updating the file system using Mender, the tool will overwrite the root file system partition that is not currently mounted (note: the onboard flash storage contains two separate root file system partitions, only one is ever used at a single time). Any data stored on that partition will be permanently lost, including the currently loaded FPGA image. After updating that partition, it will reboot into the newly updated partition. Only if the update is confirmed by the user, the update will be made permanent. This means that if an update fails, the device will be always able to reboot into the partition from which the update was originally launched (which presumably is in a working state). Another update can be launched now to correct the previous, failed update, until it works.

To obtain the file system Mender image (these are files with a `.mender` suffix), run the following command on the host computer with Internet access:

```
$ sudo uhd_images_downloader -t mender -t x4xx --yes
```

NOTE: In the output of the command, the folder destination where the images are saved is printed out.

Next, you will need to copy this Mender file system image to the USRP X4xx. This can be done with the Linux utility `scp`.

```
$ scp /usr/local/share/uhd/images/usrp_x4xx_fs.mender root@192.168.1.51:~/.
```

Note: The path and IP may different for your configuration, the command above assumes you're using the default installation path of `/usr/local` and that the X4xx's IP is `192.168.1.51`.

After copying the Mender file system image to the X4xx, connect to the X4xx using either the Serial Console, or via SSH to gain shell access.

On the X4xx, run `mender install /path/to/latest.mender` to update the file system:

```
$ mender install /home/root/usrp_x4xx_fs.mender
```

The artifact can also be stored on a remote server:

```
$ mender install http://server.name/path/to/latest.mender
```

This procedure will take a few minutes to complete. After mender has logged a successful update, reboot the device:

```
$ reboot
```

If the reboot worked, and the device seems functional, commit the changes so that the boot loader knows to permanently boot into this partition:

```
$ mender commit
```

To identify the currently installed Mender artifact from the command line, the following file can be queried on the X4x0:

```
$ mender show-artifact
```

If you are using a Mender server, the updates can be initiated from a web dashboard. From there, you can start the updates without having to log into the device, and you can update groups of USRPs with a few clicks in a web GUI. The dashboard can also be used to inspect the state of USRPs. This is a simple way to update groups of rack-mounted USRPs with custom file systems.

If you are running a hosted server, the updates can be initiated from a web dashboard. From there, you can start the updates without having to log into the device, and can update groups of USRPs with a few clicks in a web GUI. The dashboard can also be used to inspect the state of USRPs. This is a simple way to update groups of rack-mounted USRPs with custom file systems.

The Ettus USRP X4x0 has various network interfaces:

eth0: RJ45 port.

The RJ45 port comes up with a default configuration of DHCP, that will request a network address from your DHCP server (if available on your network). This interface is agnostic of FPGA image flavor.

int0: internal interface for network communication between the embedded ARM processor and FPGA.

The internal network interface is configured with a static address: 169.254.0.1/24. This interface is agnostic of FPGA image flavor.

sfpX [, sfpX_1, sfpX_2, sfpX_3]: QSFP28 network interface(s), up-to four (one per lane) based on implemented protocol.

Each QSFP28 port has four high-speed transceiver lanes. Therefore, depending on the FPGA image flavor, up-to four different network interfaces may exist per QSFP28 port, using the sfpXfor the first lane, and sfpX_1-3 for the other three lanes. Each network interface has a default static IP address. Note that for multi-lane protocols, such as 100 GbE, a single interface is used (sfpX). The configuration files for these network interfaces are stored in: `/data/network/`

Interface Name	Description	Default Configuration	Configuration File	Example: X4_200/X4_400 FPGA image
eth0	RJ45	DHCP	eth0.network	DHCP
int0	Internal	169.254.0.1/24	int0.network	169.254.0.1/24
sfp0	QSFP28 0 (4-lanes interface or lane 0)	192.168.10.2/24	sfp0.network	192.168.10.2/24
sfp0_1	QSFP28 0 (lane 1)	192.168.11.2/24	sfp0_1.network	192.168.11.2/24
sfp0_2	QSFP28 0 (lane 2)	192.168.12.2/24	sfp0_2.network	192.168.12.2/24
sfp0_3	QSFP28 0 (lane 3)	192.168.13.2/24	sfp0_3.network	192.168.13.2/24
sfp1	QSFP28 1 (4-lanes interface or lane 0)	192.168.20.2/24	sfp1.network	N/C
sfp1_1	QSFP28 1 (lane 1)	192.168.21.2/24	sfp1_1.network	N/C
sfp1_2	QSFP28 1 (lane 2)	192.168.22.2/24	sfp1_2.network	N/C
sfp1_3	QSFP28 1 (lane 3)	192.168.23.2/24	sfp1_3.network	N/C

Once the X4x0 has booted, determine the IP address and verify network connectivity by running uhd_find_devices on the host computer:

X410:

```
$ uhd_find_devices

-- UHD Device 0

Device Address:
serial: 1234ABC
addr: 10.2.161.10
claimed: False
mgmt_addr: 10.2.161.10
product: x410
type: x4xx
```

X440:

```
$ uhd_find_devices

-- UHD Device 0

Device Address:
serial: 1234ABC
addr: 10.2.161.10
claimed: False
mgmt_addr: 10.2.161.10
product: x440
type: x4xx
```

By default, an X4x0 will use DHCP to attempt to find an address.

At this point, you should run:

`uhd_usrp_probe --args addr=<IP address>` to ensure functionality of the device.

Note: If you receive the following error:

`Error: RuntimeError: Graph edge list is empty for rx channel 0` then you will need to download a UHD-compatible FPGA as described in Updating the FPGA or using the following command (it assumes that FPGA images have been downloaded previously using uhd_images_downloader, or that the command is run on the device itself):

X410: `uhd_image_loader --args type=x4xx,addr=<ip address>,fpga=X4_200`

X440: `uhd_image_loader --args type=x4xx,addr=<ip address>,fpga=X4_400`

When running on the device, use `127.0.0.1` as the IP address.

You can now use existing UHD examples or applications (such as rx_sample_to_file, rx_ascii_art_dft, or tx_waveforms) or other UHD-compatible applications to start receiving and transmitting with the device.

See Network Interfaces for further details on the various network interfaces available on the X4x0.

The Ettus USRP X4x0 is equipped with status LEDs for its network-capable ports: RJ45 and QSFP28s, see RJ45 LED Behavior and QSFP28 LED Behavior accordingly.

The RJ45 port has two independent LEDs: green (right) and yellow (left). The table below summarizes the LEDs' behavior. Note that link speed indication is not currently supported.

Link / Activity	**Green LED**	**Yellow LED**
No Link	Off	Off
Link / No Activity	On	Off
Link / Activity	On	Blinking

Each QSFP28 connector has four LEDs, one for each high-speed transceiver lane. The table below summarizes the LEDs' behavior, note that for multi-lane protocols, such as 100 GbE, the corresponding LEDs are ganged together. Within the same image, multiple speeds on the same port (e.g., both 10 GbE and 100 GbE) are not supported, therefore link speed indication is not supported.

Link / Activity	**QSFP28 LED (4 Total)**
No Link	Off
Link / No Activity	Green (solid)
Link / Activity	Amber (blinking)

The X4x0 ships without a root password set. It is possible to ssh into the device by simply connecting as root, and thus gaining access to all subsystems. To set a password, run the command

`$ passwd` on the device.

It is possible to gain access to the device using a serial terminal emulator. To do so, the USB debug port needs to be connected to a separate computer to gain access. Most Linux, OSX, or other Unix flavors have a tool called 'screen' which can be used for this purpose, by running the following command:

`$ sudo screen /dev/ttyUSB2 115200` In this command, we prepend 'sudo' to elevate user privileges (by default, accessing serial ports is not available to regular users), we specify the device node (in this case, /dev/ttyUSB2), and the baud rate (115200).

The exact device node depends on your operating system's driver and other USB devices that might be already connected. Modern Linux systems offer alternatives to simply trying device nodes; instead, the OS might have a directory of symlinks under /dev/serial/by-id:

```
$ ls /dev/serial/by-id
usb-Digilent_Digilent_USB_Device_2516351DDCC0-if02-port0
usb-Digilent_Digilent_USB_Device_2516351DDCC0-if03-port0
```

Note: Exact names depend on the host operating system version and may differ.

The first (with the if02 suffix) connects to the STM32 microcontroller (SCU), whereas the second (with the if03 suffix) connects to Linux running on the RFSoC APU.

`$ sudo screen /dev/serial/by-id/usb-Digilent_Digilent_USB_Device_2516351DDCC0-if03-port0 115200` After entering the username root (no password is set by default), you should be presented with a shell prompt similar to the following:

`root@ni-x4xx-1234ABC:~#` On this prompt, you can enter any Linux command available. Using the default configuration, the serial console will also show all kernel log messages (unlike when using SSH, for example), and give access to the boot loader (U-boot prompt). This can be used to debug kernel or bootloader issues more efficiently than when logged in via SSH.

The microcontroller (which controls the power sequencing, among other things) also has a serial console available. To connect to the microcontroller, use the other UART device. In the example above:

```
$ sudo screen /dev/serial/by-id/usb-Digilent_Digilent_USB_Device_2516351DDCC0-if02-port0 115200
```

It provides a very simple prompt. The command 'help' will list all available commands. A direct connection to the microcontroller can be used to hard-reset the device without physically accessing it and other low-level diagnostics. For example, running the command reboot will emulate a reset button press, resetting the state of the device, while the command powerbtn will emulate a power button press, turning the device back on again.

The USRP X4x0 has two network connections: The dual QSFP28 ports, and an RJ45 connector. The latter is by default configured by DHCP; by plugging it into into 1 Gigabit switch on a DHCP-capable network, it will get assigned an IP address and thus be accessible via ssh.

In case your network setup does not include a DHCP server, refer to the section Serial Connection. A serial login can be used to assign an IP address manually.

After the device obtained an IP address you can log in from a Linux or OSX machine by typing:

`$ ssh root@ni-x4xx-1234ABC # Replace with your actual device name!` Depending on your network setup, using a .local domain may work:

`$ ssh root@ni-x4xx-1234ABC.local` Of course, you can also connect to the IP address directly if you know it (or set it manually using the serial console).

Note: The device's hostname is derived from its serial number by default (`ni-x4xx-$SERIAL`). You can change the hostname by creating the file `/data/network/hostname`, saving the desired hostname in it, then rebooting.

On Microsoft Windows, the connection can be established using a tool such as PuTTY, by selecting a username of root without password.

Like with the serial console, you should be presented with a prompt like the following:

`root@ni-x4xx-1234ABC:~#`

The USRP X4x0 can be configured to power on and boot automatically when power is applied. This setting can be controlled using the `eeprom-set-autoboot` script. This script is executed directly on the USRP X4x0. To enable autoboot, run `eeprom-set-autoboot on`; to disable autoboot, run `eeprom-set-autoboot off`.

The FPGA can be updated simply using uhd_image_loader:

`uhd_image_loader --args type=x4xx,addr=<IP address of device> --fpga-path <path to .bit>` or

`uhd_image_loader --args type=x4xx,addr=<IP address of device>,fpga=FPGA_TYPE` A UHD install will likely have pre-built images in /usr/share/uhd/images/. Up-to-date images can be downloaded using the uhd_images_downloader script:

`uhd_images_downloader` will download images into /usr/share/uhd/images/ (the path may differ, depending on how UHD was installed).

Also note that the USRP already ships with compatible FPGA images on the device - these images can be loaded by SSH'ing into the device and running:

X410: `uhd_image_loader --args type=x4xx,mgmt_addr=127.0.0.1,fpga=X4_200`

X440: `uhd_image_loader --args type=x4xx,mgmt_addr=127.0.0.1,fpga=X4_400`

Unlike the USRP X310 or other third-generation USRP devices, the FPGA image flavors do not only encode how the QSFP28 connectors are configured, but also which master clock rates are available. This is because the data converter configuration is part of the FPGA image (the ADCs/DACs on the X4x0 are on the same die as the FPGA). The image flavors consist of two short strings, separated by an underscore, e.g. X4_200 (X410) or X4_400 (X440) is an image flavor which contains 4x 10 GbE, and can handle an analog bandwidth of 200 MHz or 400 MHz respectively. The first two characters describe the configuration of the QSFP28 ports: 'X' stands for 10 GbE, 'C' stands for 100 GbE. For details see FPGA Image Flavor in the USRP Hardware Driver and USRP Manual.

The analog bandwidth determines the available master clock rates.

X410: As of UHD 4.1, only the X4_200 image is shipped with UHD, which allows a 245.76 MHz or 250 MHz master clock rate. With UHD 4.2, the CG_400 image was added allowing for 491.52 MHz and 500 MHz master clock rates. With UHD 4.5, the UC_200 image (245.76 MHz and 250 MHz master clock rate) was added.

X440: As of UHD 4.5, UHD ships with X4_400, X4_1600, CG_400 and CG_1600 images. The X4_400 and CG_400 images allow master clock rates between 125 MHz and 512 MHz and the usage of all 8 channels while the X4_1600 and CG_1600 images allow master clock rates between 125 MHz and 2048 MHz but only the usage of channels 0 and 4.

Any other images are considered experimental (unsupported).

Key	Description	Example Value
addr	IPv4 address of primary SFP+ port to connect to.	addr=192.168.30.2
second_addr	IPv4 address of secondary SFP+ port to connect to.	second_addr=192.168.40.2
mgmt_addr	IPv4 address or hostname to which to connect the RPC client. Defaults to `addr'.	mgmt_addr=ni-sulfur-311FE00
find_all	When using broadcast, find all devices, even if unreachable via CHDR.	find_all=1
master_clock_rate	Master Clock Rate in Hz.	master_clock_rate=250e6
converter_rate	Converter Rate in Hz. Only X440 and together with master_clock_rate.	master_clock_rate=250e6,converter_rate=1000e6
serialize_init	Force serial initialization of daughterboards.	serialize_init=1
skip_init	Skip the initialization process for the device.	skip_init=1
time_source	Specify the time (PPS) source.	time_source=internal
clock_source	Specify the reference clock source.	clock_source=internal
ref_clk_freq	Specify the external reference clock frequency, default is 10 MHz.	ref_clk_freq=20e6
discovery_port	Override default value for MPM discovery port.	discovery_port=49700
rpc_port	Override default value for MPM RPC port.	rpc_port=49701

This is only a subset of the existing device arguments. For a complete list please consult the UHD user manual of the X4x0 device series.

The USRP X4x0 includes a Jackson Labs LTE-Lite GPS module. Its antenna port is on the rear panel. When the X4x0 has access to GPS satellite signals, it can use this module to read out the current GPS time and location as well as to discipline an onboard OCXO.

To use the GPS as a clock and time reference, set the device arguments `time_source` and `clock_source` to `gpsdo`.

Note the GPS module is not enabled when the clock source is not set to `gpsdo`.

Its power-on status can be queried using the `gps_enabled` GPS sensor. When disabled, none of the sensors will return useful (if any) values.

Note that acquiring a GPS lock can take some time after enabling the GPS, so if a UHD application is enabling the GPS dynamically, it might take some time before a GPS lock is reported.

To set the clock source and time source dynamically, see the following code:

```
// Set clock/time individually:
usrp->set_clock_source("gpsdo");
usrp->set_time_source("gpsdo");
// This is equivalent to the previous commands, but faster, as it sets
// both settings simultaneously and avoids duplicating settings that are shared
// between these calls.
usrp->set_sync_source("clock_source=gpsdo,time_source=gpsdo");
```

The USRP X4x0 has two HDMI front-panel connectors, which are connected to the FPGA. For a description of the GPIO control API, see the USRP X4x0 GPIO UHD Manual Entry, the USRP X4x0 Series Manual, the ZBX ATR section (X410) and the FBX ATR section (X440).

The RF ports on the front panel of the X410 + ZBX correspond to the following subdev specifications:

Label	Subdev Spec
DB 0 / RF 0	A:0
DB 0 / RF 1	A:1
DB 1 / RF 0	B:0
DB 1 / RF 1	B:1

The RF ports on the front panel of the X440 + FBX correspond to the following subdev specifications (for xx_400 FPGA images):

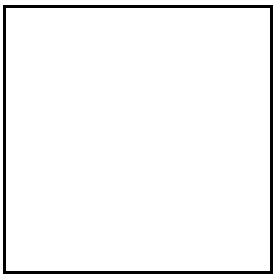
Label	Subdev Spec
DB 0 / RF 0	A:0
DB 0 / RF 1	A:1
DB 0 / RF 2	A:2
DB 0 / RF 3	A:3
DB 1 / RF 0	B:0
DB 1 / RF 1	B:1
DB 1 / RF 2	B:2
DB 1 / RF 3	B:3

When using a xx_1600 FPGA image on X440, only A:0 and B:0 are available.

The subdev spec slot identifiers "A" and "B" are not reflected on the front panel. They were set to match valid subdev specifications of previous USRPs, maintaining backward compatibility.

These values can be used for uhd::usrp::multi_usrp::set_rx_subdev_spec() and uhd::usrp::multi_usrp::set_tx_subdev_spec() as with other USRPs.

The USRP X4x0 is equipped with four LEDs located on the device's rear panel. Each LED supports four different states: Off, Green, Red, and Amber. One LED (PWR) indicates the device's power state (see Power LED below). The other three LEDs (LED 0, LED 1, and LED 2) are user-configurable, different behaviors are supported for each of these LEDs (see User-configurable LEDs below).



Power LED The USRP X4x0's PWR LED is reserved to visually indicate the user the device's power state. Power LED Behavior describes what each LED state represents.

PWR LED State	Meaning
Off	No power is applied
Amber	Power is good but X4x0 is powered off
Green	Power is good and X4x0 is powered on
Red	Power error state

The USRP X4x0's user-configurable rear panel status LEDs (LED 0, LED 1, and LED 2) allow the user to have visual indication of various device conditions. Supported LED Behaviors provides a complete list of the supported behaviors for each user-configurable LED. By default, these LEDs are configured as described in LEDs Default Behavior.

The user may alter the default LEDs behavior either temporarily or persistently, see the Temporarily change the LED Behavior or Persistently in the UHD manual to change the LED Behavior accordingly.