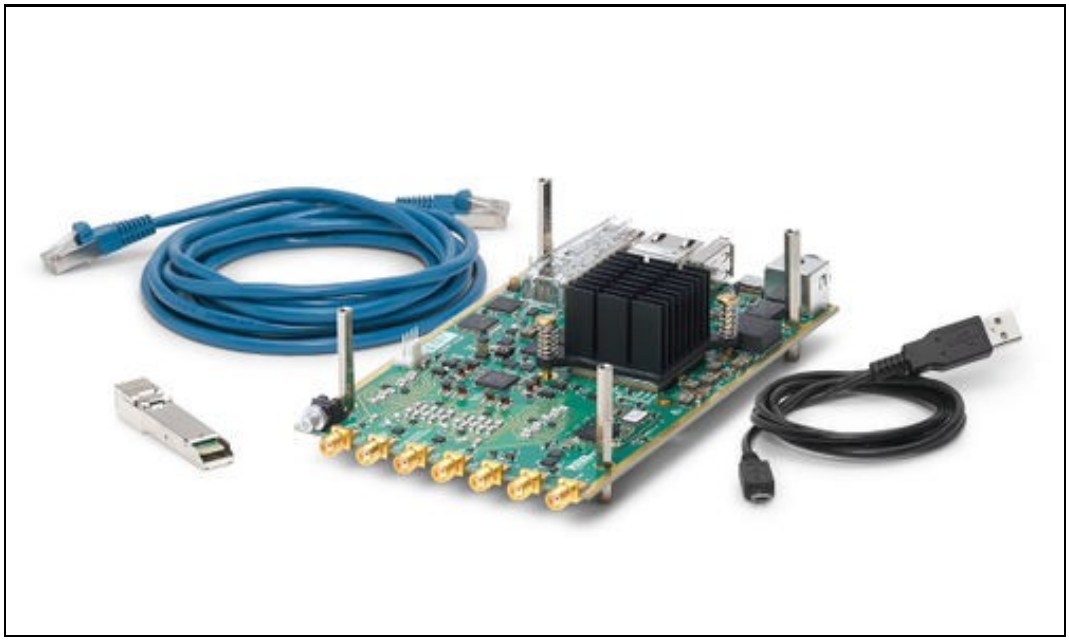
E320 Getting Started Guide

Contents

- USRP E320
- Power connector (assembly required)
- 4 M3x0.5, M3x5 Standoffs
- 1 Gb Ethernet Cat-5e Cable (3m)
- USB-A to Micro USB-B Cable (1m)
- 1 Gb SFP+ to RJ45 Adapter
- Getting Started Guide
- Ettus Research Sticker



- USRP E320 in enclosure
- DC Power Supply (12V, 7A)
- 1 Gb Ethernet Cat-5e Cable (3m)
- USB-A to Micro USB-B Cable (1m)
- 1 Gb SFP+ to RJ45 Adapter
- Getting Started Guide
- Ettus Research Sticker
- T8 Torx Wrench

Ensure that your kit contains all the items listed above. If any items are missing, please contact sales@ettus.com? immediately.

- For Network Mode: A host computer with an 1 or 10 Gb Ethernet interface. If operating with the 10 Gb Ethernet interface, the "XG" FPGA image must be loaded before the SFP+ port will operate at 10 Gb speeds. Optionally a second 1 Gb Ethernet interface can be used to connect to the onboard ARM CPU for remote management.
- For Embedded Mode: A host computer is only required for initial device configuration, remote control and management, or data visualization. The host computer can connect to the RJ45 1 Gb port or Serial Console port to remotely access the Open Embedded Linux operating system running on the ARM CPU. Once configured, the USRP E320 can operate as a stand-alone device without a connection to a remote host computer.
- For Board-only Version: A third-party 10-14V/3A power supply, which requires assembly with the power connect components included in the kit. An assembled power supply can be purchased here: https://www.ettus.com/product/details/12V-PWR

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 anything especially metal objects to touch the board while it is powered on.
- 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.
- Never touch the circuit board or heatsink while the device is powered on.
- All connections should be made/removed while is device is powered off.

 Never apply more than -15 dBm of power into any RF input.

 Always use at least 30dB attenuation if operating in loopback configuration

To use your Universal Software Radio Peripheral (USRP?), you must have software tools correctly installed and configured on your host computer. Step-by-step guides for these software tools are found in the Application Notes for Building and Installing the USRP Open-Source Toolchain (UHD and GNU Radio) on Linux, OS X and Windows.

The USRP E320 requires UHD version 3.13.0.2 or later. It is strongly? recommended to use the latest stable release of UHD on both the host computer and the USRP via the filesystem on the SD card. If this release fails to work in some way, then try the maintenance branch of the latest stable version. If you are operating the device in Network Mode, the version of UHD running on the host machine and E320 USRP must match to within the same maintenance release and branch. See the UHD GitHub repository for the latest release and maintenance branch.

Listed below are the interfaces to connect to the USRP E320. Each interface has specific functionality, limitations and purpose.

Serial Console

The Serial Console provides a low-level interface to the ARM CPU and STM32 microcontroller, typically used for debugging. The serial console can also be used as a JTAG connection to the FPGA.

1 Gb RJ45 Connection

The 1 Gb RJ45 Connection interfaces with the on-board ARM CPU. When operated in "Network mode", this interface can optionally be used for remote control and management traffic. Regardless of the operation mode (Host vs Embedded) this interface can be used to connect to the ARM via SSH. By default, the 1 Gb RJ45 connection is configured to use a DHCP assigned IP address.

SFP+ Connection

The SFP+ Connection supports multiple interfaces for streaming high-speed, low-latency data, depending upon which FPGA image is loaded.

It is possible to gain shell access to the device using a serial terminal emulator via the Serial Console port. Most Linux, OS X, or other Unix based operating systems have a utility called `screen` which can be used for this purpose.

If you do not have `screen` installed, it can be installed via your distribution's package manager. For Ubuntu/Debian based operating systems it can be installed with the package manager `apt` such as:

```
sudo apt install screen
```

The default Baud Rate for the Serial Console is: `115200`

The exact device node you should attach to depends on your operating system's driver and other USB devices that might already be 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-FTDI_Dual_RS232-HS-if00-port0
usb-FTDI_Dual_RS232-HS-if01-port0
usb-Silicon_Labs_CP2105_Dual_USB_to_UART_Bridge_Controller_007F6A69-if00-port0
usb-Silicon_Labs_CP2105_Dual_USB_to_UART_Bridge_Controller_007F6A69-if01-port0
```

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

Every E320 series device connected to USB will by default show up as four different devices. The devices labeled `"USB_to_UART_Bridge_Controller"` are the devices that offer a serial prompt. The first (with the `if00` suffix) connects to the `STM32 Microcontroller`, whereas the second connects to the `ARM CPU`.

If you have multiple E320 Serial Consoles connected to a single host, you may have to empirically test nodes.

Connecting to the ARM CPU can be performed with the command:

```
$ sudo screen  /dev/serial/by-id/usb-Silicon_Labs_CP2105_Dual_USB_to_UART_Bridge_Controller_007F6A69-if01-port0 115200
```

Upon starting the USRP E320, boot messages will appear and rapidly update. Once the boot process successfully completes, a login prompt like the following should appear:

```
Alchemy 2018.04 ni-e320-serial ttyPS0
ni-e320-serial login:
```

Enter the username: ?`root`?

By default, the `root` user's password is left blank. Press the `Enter` key when prompted for a password.

You should now be presented with a shell prompt similar to the following:

```
root@ni-e320-<motherboard serial #>:~#
```

Using the default configuration, the serial console will show all kernel log messages (which are not available when using SSH) and give access to the boot loader (U-boot prompt). This can be used to debug kernel or boot-loader issues more efficiently than when logged in via SSH.

Using the Serial Console interface, it is possible to connect to the STM32 microcontroller with the command below. The STM32 controls the power sequencing and several other low-level device operations.

```
$ sudo screen  /dev/serial/by-id/usb-Silicon_Labs_CP2105_Dual_USB_to_UART_Bridge_Controller_007F6A69-if00-port0 115200
```

The STM32 interface 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 (i.e., emulating a power button press) and other low-level diagnostics.

By default, the RJ45 1 Gb management interface is configured to be assigned a DHCP IP address.

If you have access to a network which provides a DHCP server (such as a common router's LAN), attach the RJ45 1 Gb port to this network. Details vary by vendor, however, most router management interfaces will provide a list of attached devices to the LAN including their IP address.

Without access to a router management interface, you can identify the IP address by connecting to the ARM CPU via Serial Console as detailed in the section above and running the command `ip a`:

Example Output:

```
# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.151/24 brd 192.168.1.255 scope global dynamic eth0
       valid_lft 42865sec preferred_lft 42865sec
3: sfp0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 8000 qdisc pfifo_fast qlen 1000
    link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
    inet 192.168.10.2/24 brd 192.168.10.255 scope global sfp0
       valid_lft forever preferred_lft forever
```

If you do not have access to a network with a DHCP server, you can create one using the Linux utility `dnsmasq`:

```
$ sudo dnsmasq -i <ETHERNET_ADAPTER_NAME> --dhcp-range=192.168.1.50,192.168.1.100 --except-interface=lo --bind-dynamic --no-daemon
```

NOTE: Modify the value `<ETHERNET_ADAPTER_NAME>` to match the interface you would like to create a DHCP server on.

After the device has obtained an IP address, you can remotely log into it from a Linux or macOS systems with SSH, as shown below:

```
$ ssh root@192.168.1.51
```

NOTE: The IP address may vary depending on your network setup.

NOTE: The `root` password is empty/blank.

On Microsoft Windows, the SSH connection can be established using the third-party program, such as ?PuTTY.

After logging in, you should be presented with a shell prompt like the following:

```
root@ni-e320-<motherboard serial #>:~#
```

Before operating the device, it is? ?strongly? recommended to update to the latest version of the Embedded Linux file system. If you are operating the device in Network Mode, the version of UHD running on the host machine and E320 USRP must match.

There is two ways to update the file system for the E320 USRP:

1. Mender

2. Physically remove microSD card from device and write a new file system to the microSD card.

The SD Card is divided into four partitions. There are two root file system partitions, a "boot" partition and a "data" partition.

Any data you would like to preserve through Mender updates should be saved to the "data" partition, which is mounted at `/data`.

Mender is third-party software that enables remote updating of the root file system without physically accessing the device (see also the Mender website https://mender.io). 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. Users can host their own local Mender server, or use servers hosted by Mender as a paid service; contact Mender for more information.

When updating the file system using Mender, the tool will overwrite the root file system partition that is not currently mounted. Any data stored in the root partitions will be permanently lost with a Mender update.

After updating a partition with Mender, 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 e320 --yes
```

Example Output:

```
[INFO] Using base URL: https://files.ettus.com/binaries/cache/
[INFO] Images destination: /usr/local/share/uhd/images
[INFO] No inventory file found at /usr/local/share/uhd/images/inventory.json. Creating an empty one.
301483 kB / 301483 kB (100%) e3xx_e320_mender_default-v4.4.0.0.zip
[INFO] Images download complete.
```

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 E320. This can be done with the Linux utility `scp`.

```
$ scp /usr/local/share/uhd/images/usrp_e320_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 E320's IP is `192.168.1.51`.

After copying the Mender file system image to the E320, connect to the E320 using either the Serial Console, or via SSH to gain shell access.

On the E320, run `mender install /path/to/latest.mender` to update the file system:

```
root@ni-e320-serial:~# mender install /home/root/usrp_e320_fs.mender
```

Example Output:

```
root@ni-e320-316E375:~# mender install /home/root/usrp_e320_fs.mender
INFO[0000] Start updating from local image file: [/home/root/usrp_e320_fs.mender]  module=rootfs
Installing update from the artifact of size 399640064
INFO[0000] opening device /dev/mmcblk0p3 for writing     module=block_device
INFO[0000] partition /dev/mmcblk0p3 size: 2046820352     module=block_device
................................   0% 1024 KiB
................................   0% 2048 KiB
................................   0% 3072 KiB
[truncated for readability]
................................  99% 389120 KiB
................................  99% 390144 KiB
................................ 100% 390273 KiB
INFO[0740] wrote 2046820352/2046820352 bytes of update to device /dev/mmcblk0p3  module=device
INFO[0744] Enabling partition with new image installed to be a boot candidate: 3  module=device
```

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 E320:

```
$ cat /etc/mender/artifact_info
```

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.

For more information on updating the file-system, refer to the E3xx page in the Devices section of the UHD Manual at ?http://uhd.ettus.com.

The microSD card is accessible directly on the Board-only version of the E320 USRP. The E320 Full Enclosure version must be opened with the included Torx wrench.

NOTE: This method will overwrite all data saved on the microSD card, including any data saved to the `/data` partition.

Please see the separate application note, Writing the USRP File System Disk Image to a SD Card, for step-by-step instructions on writing the file system image to the microSD card.

The USRP E320 systemd network configuration files are located either at: `/etc/systemd/network/`

```
# ls /etc/systemd/network/
eth0.network  sfp0.network
```

or for newer versions of the file system: `/data/network/`

```
# ls /data/network/
eth0.network  int0.network  sfp0.network
```

For details on configuration please refer to the systemd-networkd manual pages.

The factory settings are as follows:

```
eth0 (DHCP):

    [Match]
    Name=eth0

    [Network]
    DHCP=v4

    [DHCPv4]
    UseHostname=false

sfp0 (static):

    [Match]
    Name=sfp0

    [Network]
    Address=192.168.10.2/24

    [Link]
    MTUBytes=8000
```

Additional notes on networking:

- Care needs to be taken when editing these files on the device, since `vi` / `vim` sometimes generates undo files (e.g. `/data/network/sfp0.network~`), that `systemd-networkd` might accidentally pick up.
- Temporarily setting the IP addresses or MTU sizes via `ifconfig` or other command line tools will only change the value until the next reboot or reload of the FPGA image.
- If the MTU of the device and host computers differ, streaming issues can occur.
- Streaming via SFP0 at 1 Gb rates requires a MTU of `1500`
- Streaming via SFP0 at 10 Gb rates requires a MTU of `8000`

For addition details on network configuration here: https://files.ettus.com/manual/page_usrp_e320.html#e320_network_configuration

The FPGA image should match the version of UHD installed on the host computer when operated in Network mode.

Network mode FPGA image updates must be made through the RJ45 management interface.

To obtain all the FPGA images for your installed version of UHD, run the following command on the host computer with internet access:

```
$ sudo uhd_images_downloader -t e320 -t fpga
```

Example Output:

```
$ uhd_images_downloader -t e320 -t fpga
[INFO] Images destination: /usr/local/share/uhd/images
[INFO] No inventory file found at /usr/local/share/uhd/images/inventory.json. Creating an empty one.
05920 kB / 05920 kB (100%) e3xx_e320_fpga_default-g494ae8bb.zip
[INFO] Images download complete.
```

There is two versions of the E320 FPGA images shipped with UHD:

- `1G` for 1 Gb rates on the SFP+ port (default image)

- `XG` for 10 Gb rates on the SFP+ port

In this example, we load the `XG` variant of the FPGA image.

```
$ uhd_image_loader --args "type=e3xx,mgmt_addr=<E320_RJ45_IP_ADDR>,fpga=XG"
```

Example Output:

```
$ uhd_image_loader --args "mgmt_addr=192.168.1.51,type=e3xx,fpga=XG"
[INFO] [UHD] linux; GNU C++ version 5.4.0 20160609; Boost_105800; UHD_3.13.1.0-1-gd3b7e90a
[INFO] [MPMD] Initializing 1 device(s) in parallel with args: mgmt_addr=192.168.1.51,type=e3xx,product=e320,serial=316E375,claimed=False,s
[INFO] [MPMD] Claimed device without full initialization.
```

```
[INFO] [MPMD IMAGE LOADER] Starting update. This may take a while.
[INFO] [MPM.PeriphManager] Updating component `fpga'
[INFO] [MPM.PeriphManager] Updating component `dts'
[INFO] [MPM.RPCServer] Resetting peripheral manager.
[INFO] [MPM.PeriphManager] Device serial number: 316E375
[INFO] [MPMD IMAGE LOADER] Update component function succeeded.
[INFO] [MPM.PeriphManager] Found 1 daughterboard(s).
```

The FPGA is immediately updated, and this FPGA image will continue to be used. The device does not need to be power cycled to use the new image.

To load a different FPGA image (i.e. `1G`), modify the device argument `fpga=` to a value of `fpga=1G`.

To specify the path to a custom FPGA image, use the ?`--fpga-path`? argument.

```
$ uhd_image_loader --args "type=e3xx,mgmt_addr=<E320_RJ45_IP_ADDR>" --fpga-path=/path/to/custom/fpga.bit
```

The Verilog code for the FPGA in the USRP E320 is open-source, and users are free to modify and customize it for their needs. However, certain modifications may result in either bricking the device, or even in physical damage to the unit. Please note that modifications to the FPGA are made at the risk of the user, and may not be covered by the warranty of the device.

It is possible to update the FPGA image when operated in Embedded mode. Connect to the ARM CPU via Serial Console or via SSH. It is generally recommend to use SSH over the RJ45 interface for remote management.

Run the command `uhd_images_downloader` to download the FPGA images to the device's file system:

NOTE: The 1 Gb RJ45 management interface will require Internet access for this next step.

```
root@ni-e320-serial:~# python3 /usr/bin/uhd_images_downloader -t e320 -t fpga
[INFO] Images destination: /usr/share/uhd/images
[INFO] No inventory file found at /usr/share/uhd/images/inventory.json. Creating an empty one.
05920 kB / 05920 kB (100%) e3xx_e320_fpga_default-g494ae8bb.zip
[INFO] Images download complete.
```

NOTE: The default UHD FPGA Images destination within the E320's file-system is `/usr/share/uhd/images`. The default UHD FPGA Images destination on a typical host installation is `/usr/local/share/uhd/images`.

Updating the FPGA image from the ARM CPU is the same as detailed above for a Network mode update:

```
root@ni-e320-serial:~# uhd_image_loader --args "type=e3xx,fpga=1G"
[INFO] [UHD] linux; GNU C++ version 7.3.0; Boost_106600; UHD_3.13.1.0-0-unknown
[INFO] [MPMD] Initializing 1 device(s) in parallel with args: mgmt_addr=127.0.0.1,type=e3xx,product=e320,serial=316E375,claimed=False,skip_in
[INFO] [MPM.PeriphManager.UDP] No CHDR interfaces found!
[INFO] [MPM.PeriphManager.UDP] No CHDR interfaces found!
[INFO] [MPMD] Claimed device without full initialization.
[INFO] [MPMD IMAGE LOADER] Starting update. This may take a while.
[INFO] [MPM.PeriphManager] Updating component `fpga'
[INFO] [MPM.PeriphManager] Updating component `dts'
[INFO] [MPM.RPCServer] Resetting peripheral manager.
[INFO] [MPM.PeriphManager] Device serial number: 316E375
[INFO] [MPMD IMAGE LOADER] Update component function succeeded.
[INFO] [MPM.PeriphManager] Found 1 daughterboard(s).
```

For more information on updating the FPGA image, refer to the UHD Manual at http://uhd.ettus.com.

The device supports multiple high-speed, low-latency interfaces on the SFP+ port for streaming samples to the host computer.

Complete the steps below to set up a streaming connection over the 1 Gb Ethernet interface on the `SFP+ Port`.

NOTE: The `1G` FPGA image must be loaded for the `SFP+ Port` to operate at 1 Gb speeds. If the `XG` image is loaded, the port will be unresponsive at 1Gb speeds.

1. Configure your Host's 1 Gb Ethernet interface as shown below. This interface should be separate from the 1 Gb NIC/network which is connected to the 1 Gb RJ45 management interface.

```
IP Address: 192.168.10.1
Subnet Mask: 255.255.255.0
Gateway: 0.0.0.0
MTU: 1500
```

NOTE: When operating the `SFP+ Port` at 1 Gb speeds, it is important to set a MTU of `1500` and not a value of `automatic`. Mismatched MTU values on either the Host or E320 may cause flow control errors. Your computer may need to be restarted for the MTU value to take effect.

2. Insert the RJ45-to-SFP+ adapter ?into the? `SFP+ Port`?.

3. Connect the SFP+ adapter on the device to an Ethernet port on the host computer using a standard Ethernet cable.

The ? Green LED? above the ?`SFP+ Port`? should illuminate.

4. To test the connection,? ?`ping`? the device at address `192.168.10.2?` from the host, as shown below:

```
$ ping 192.168.10.2
PING 192.168.10.2 (192.168.10.2) 56(84) bytes of data.
64 bytes from 192.168.10.2: icmp_seq=1 ttl=64 time=1.06 ms
^C
--- 192.168.10.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.065/1.065/1.065/0.000 ms
```

Press `CTRL+C` to stop the ping program.

5. Verify your MTU is set correctly for 1 Gb speeds on the E320. See the section Updating the Network Configurations for additional details.

Proceed to the next section Verifying Device Operation.

Load the `XG` FPGA image for 10 Gb streaming as detailed in the section Updating the FPGA Image. You will need to use a 10 GigE cable that can be plugged in directly to the SFP+ connector on the board.

NOTE: The `XG` FPGA image must be loaded for the `SFP+ Port` to operate at 10 Gb speeds. If the `1G` image is loaded, the port will be unresponsive at 10 Gb speeds. Mismatched MTU values on either the Host or E320 may cause flow control errors.

1. Configure your Host's 10 Gb Ethernet interface as shown below.

```
IP Address: 192.168.10.1
Subnet Mask: 255.255.255.0
Gateway: 0.0.0.0
MTU: 8000
```

NOTE: When operating the `SFP+ Port` at 10 Gb speeds, it is important to set a MTU of `8000` and not a value of `automatic`. Mismatched MTU values on either the Host or E320 may cause flow control errors. Your computer may need to be restarted for the MTU value to take effect.

2. Connect the SFP+ port on the device to an Ethernet port on the host computer using a 10 Gb SFP+ copper or fiber cable.

The ? Green LED? above the ?`SFP+ Port`? should illuminate.

4. To test the connection,? ?`ping`? the device at address `192.168.10.2?` from the host, as shown below:

```
$ ping 192.168.10.2
PING 192.168.10.2 (192.168.10.2) 56(84) bytes of data.
64 bytes from 192.168.10.2: icmp_seq=1 ttl=64 time=1.06 ms
^C
--- 192.168.10.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.065/1.065/1.065/0.000 ms
```

Press `CTRL+C` to stop the ping program.

5. Verify your MTU is set correctly for 10 Gb speeds on the E320. See the section Updating the Network Configurations for additional details.

Proceed to the next section Verifying Device Operation.

Once you have successfully setup a management interface and streaming interface, you can now verify the devices operation using the include UHD utilities.

The USRP E320 contains 2 channels, each represented on the front panel as `RF A` and `RF B`. Below is the `subdev` mapping of RF Ports.

E320

- RF A = A:0
- RF B = A:1

Additional details of UHD Subdevice Specifications can be found here in the UHD Manual:
http://files.ettus.com/manual/page_configuration.html#config_subdev

The USRP E320 supports master clock rate from 200 kHz to 61.44 MHz and can be changed by adding `master_clock_rate=<rate>` to the default UHD args. The default master clock rate is 16 MHz.

Sample rates as delivered to/from the host computer for USRP devices are constrained to follow several important rules.

It is important to understand that strictly-integer decimation and interpolation are used within USRP hardware to meet the requested sample rate requirements of the application at hand. That means that the desired sample rate must meet the requirement that master-clock-rate/desired-sample-rate be an integer ratio. Further, it is strongly desirable for that ratio to be even. This ratio is the decimation (down-conversion) or interpolation (up-conversion) factor. The decimation or interpolation factor may be between 1 and 1024. There are further constraints on the decimation or interpolation factor. If the decimation or interpolation factor exceeds 128, then it must be evenly divisible by 2. If the decimation or interpolation factor exceeds 256, then it must be evenly divisible by 4.

Additional information on Sample Rates can be found here in the UHD Manual:
http://files.ettus.com/manual/page_general.html#general_sampleratenotes

The UHD utility `uhd_usrp_probe` provides detailed information of the USRP device.

From your host computer, run the command `uhd_usrp_probe`:

```
$ uhd_usrp_probe --args "addr=192.168.10.2"
[INFO] [UHD] linux; GNU C++ version 5.4.0 20160609; Boost_105800; UHD_3.13.1.0-1-gd3b7e90a
[INFO] [MPMD] Initializing 1 device(s) in parallel with args: mgmt_addr=192.168.10.2,type=e3xx,product=e320,serial=316E375,claimed=False,addr
[INFO] [MPM.PeriphManager] init() called with device args `product=e320,mgmt_addr=192.168.10.2'.
[INFO] [0/DmaFIFO_0] Initializing block control (NOC ID: 0xF1F0D00000000000)
[INFO] [0/DmaFIFO_0] BIST passed (Throughput: 1343 MB/s)
[INFO] [0/DmaFIFO_0] BIST passed (Throughput: 1335 MB/s)
[INFO] [0/Radio_0] Initializing block control (NOC ID: 0x12AD100000003320)
[INFO] [0/DDC_0] Initializing block control (NOC ID: 0xDDC0000000000000)
[INFO] [0/DUC_0] Initializing block control (NOC ID: 0xD0C0000000000002)
[INFO] [0/Radio_0] Performing CODEC loopback test...
[INFO] [0/Radio_0] CODEC loopback test passed
[INFO] [0/Radio_0] Performing CODEC loopback test...
[INFO] [0/Radio_0] CODEC loopback test passed
  _____________________________________________________
 /
|       Device: E300-Series Device
|     _____________________________________________________
|    /
```

```
|   |       Mboard: ni-e320-316E375
|   |   eeprom_version: 2
|   |   mpm_version: 3.13.1.0-gd3b7e90a
|   |   pid: 58144
|   |   product: e320
|   |   rev: 2
|   |   rpc_connection: remote
|   |   serial: 316E375
|   |   type: e3xx
|   |   MPM Version: 1.2
|   |   FPGA Version: 3.0
|   |   RFNoC capable: Yes
|   |
|   |   Time sources:  internal, external, gpsdo
|   |   Clock sources: external, internal, gpsdo
|   |   Sensors: gps_locked, temp_main_power, ref_locked, temp_rf_channelA, temp_fpga, gps_sky, temp_rf_channelB, fan, temp_internal, gps_tpv
|   |    _____________________________________________________
|   |   /
|   |   |       RX Dboard: A
|   |   |    _____________________________________________________
|   |   |   /
|   |   |   |       RX Frontend: 0
|   |   |   |   Name: Neon
|   |   |   |   Antennas: RX2, TX/RX
|   |   |   |   Sensors: lo_locked, ad9361_temperature, rssi, lo_lock
|   |   |   |   Freq range: 70.000 to 6000.000 MHz
|   |   |   |   Gain range PGA: 0.0 to 76.0 step 1.0 dB
|   |   |   |   Bandwidth range: 20000000.0 to 40000000.0 step 0.0 Hz
|   |   |   |   Connection Type: IQ
|   |   |   |   Uses LO offset: No
|   |   |    _____________________________________________________
|   |   |   /
|   |   |   |       RX Frontend: 1
|   |   |   |   Name: Neon
|   |   |   |   Antennas: RX2, TX/RX
|   |   |   |   Sensors: lo_locked, ad9361_temperature, rssi, lo_lock
|   |   |   |   Freq range: 70.000 to 6000.000 MHz
|   |   |   |   Gain range PGA: 0.0 to 76.0 step 1.0 dB
|   |   |   |   Bandwidth range: 20000000.0 to 40000000.0 step 0.0 Hz
|   |   |   |   Connection Type: IQ
|   |   |   |   Uses LO offset: No
|   |   |    _____________________________________________________
|   |   |   /
|   |   |   |       RX Codec: A
|   |   |   |   Name: AD9361 Dual ADC
|   |   |   |   Gain Elements: None
|   |   |    _____________________________________________________
|   |   |   /
|   |   |       TX Dboard: A
|   |   |    _____________________________________________________
|   |   |   /
|   |   |   |       TX Frontend: 0
|   |   |   |   Name: Neon
|   |   |   |   Antennas: TX/RX
|   |   |   |   Sensors: lo_locked, ad9361_temperature
|   |   |   |   Freq range: 47.000 to 6000.000 MHz
|   |   |   |   Gain range PGA: 0.0 to 89.8 step 0.2 dB
|   |   |   |   Bandwidth range: 20000000.0 to 40000000.0 step 0.0 Hz
|   |   |   |   Connection Type: IQ
|   |   |   |   Uses LO offset: No
|   |   |    _____________________________________________________
|   |   |   /
|   |   |   |       TX Frontend: 1
|   |   |   |   Name: Neon
|   |   |   |   Antennas: TX/RX
|   |   |   |   Sensors: lo_locked, ad9361_temperature
|   |   |   |   Freq range: 47.000 to 6000.000 MHz
|   |   |   |   Gain range PGA: 0.0 to 89.8 step 0.2 dB
|   |   |   |   Bandwidth range: 20000000.0 to 40000000.0 step 0.0 Hz
|   |   |   |   Connection Type: IQ
|   |   |   |   Uses LO offset: No
|   |   |    _____________________________________________________
|   |   |   /
|   |   |   |       TX Codec: A
|   |   |   |   Name: AD9361 Dual DAC
|   |   |   |   Gain Elements: None
|   |   |    _____________________________________________________
|   |   |   /
|   |   |       RFNoC blocks on this device:
|   |   |
|   |   |   * DmaFIFO_0
|   |   |   * Radio_0
|   |   |   * DDC_0
|   |   |   * DUC_0
```

The UHD driver includes several example programs, which may serve as test programs or the basis for your application program. The source code can be obtained from the UHD repository on github at: https://github.com/EttusResearch/uhd/tree/master/host/examples

You can quickly verify the operation of your USRP E320 by running the `rx_ascii_art_dft` UHD example program.

The `rx_ascii_art_dft` utility is a simple console –based, real-time FFT display tool. It is not graphical in nature, so it can be easily run over an SSH connection within a terminal window, and does not need any graphical capability, such as X Windows, to be installed. It can also be run over a serial console connection, although this is not recommended, as the formatting may not render correctly.

You can run a simple test of the E320 USRP by connecting an antenna and observing the spectrum of a commercial FM radio station in real-time, following the steps below:

1. Attach an antenna to the `RF A / RX2`– antenna port of the E320.

2. From your host computer, run the command:

```
$ /usr/local/lib/uhd/examples/rx_ascii_art_dft \
```

```
  --args "addr=192.168.10.2" \
  --freq 98.5e6 \
  --rate 2e6 \
  --gain 40 \
  --ref-lvl="-30" \
  --dyn-rng 90 \
  --ant "RX2" \
  --subdev "A:0"
```

NOTE: Modify the command−line argument `freq` ?above to specify a tuning frequency for a strong local FM radio station. You will also need to update the IP Address to match your device IP.

3. You should see a real-time FFT display of 2 MHz of spectrum, centered at the specified tuning frequency.

4. Type "`Q`" to stop the program and to return to the Linux command line.

5. You can run with the `?----help` ?argument to see a description of all available command-line options.

Example Output:

```
$ ./rx_ascii_art_dft --args "addr=192.168.10.2" --freq 98.5e6 --rate 2e6 --gain 40 --ref-lvl="-30" --dyn-rng 90 --ant "RX2" --subdev "A:0"

Creating the usrp device with: addr=192.168.10.2...
[INFO] [UHD] linux; GNU C++ version 5.4.0 20160609; Boost_105800; UHD_3.13.1.0-1-gd3b7e90a
[INFO] [MPMD] Initializing 1 device(s) in parallel with args: mgmt_addr=192.168.10.2,type=e3xx,product=e320,serial=316E375,claimed=False,addr
[INFO] [0/DmaFIFO_0] Initializing block control (NOC ID: 0xF1F0D000000000000)
[INFO] [0/DmaFIFO_0] BIST passed (Throughput: 1334 MB/s)
[INFO] [0/DmaFIFO_0] BIST passed (Throughput: 1325 MB/s)
[INFO] [0/Radio_0] Initializing block control (NOC ID: 0x12AD100000003320)
[INFO] [0/DDC_0] Initializing block control (NOC ID: 0xDDC0000000000000)
[INFO] [0/DUC_0] Initializing block control (NOC ID: 0xD0C0000000000002)
[INFO] [MPM.PeriphManager] init() called with device args `product=e320,mgmt_addr=192.168.10.2'.
[INFO] [0/Radio_0] Performing CODEC loopback test...
[INFO] [0/Radio_0] CODEC loopback test passed
[INFO] [0/Radio_0] Performing CODEC loopback test...
[INFO] [0/Radio_0] CODEC loopback test passed
Using Device: Single USRP:
  Device: E300-Series Device
  Mboard 0: ni-e320-316E375
  RX Channel: 0
    RX DSP: 0
    RX Dboard: A
    RX Subdev: Neon
  TX Channel: 0
    TX DSP: 0
    TX Dboard: A
    TX Subdev: Neon
  TX Channel: 1
    TX DSP: 1
    TX Dboard: A
    TX Subdev: Neon

Setting RX Rate: 2.000000 Msps...
Actual RX Rate: 2.000000 Msps...

Setting RX Freq: 98.500000 MHz...
Actual RX Freq: 98.500000 MHz...

Setting RX Gain: 40.000000 dB...
Actual RX Gain: 40.000000 dB...

Checking RX: all_los: locked ...

Done!
```

Included with the UHD driver example programs is a utility, `benchmark_rate` to benchmark the transport link of the system.

A system's maximum performance is dependent upon many factors. `benchmark_rate` will exercise the transport link and CPU of the system.

NOTE: This example requires the `1G` FPGA image to be loaded.

This example will test one full-duplex stream using "RFA/A:0", at a rate of 2 MS/s, for 60 seconds:

```
  /usr/local/lib/uhd/examples/benchmark_rate  \
  --args "addr=192.168.10.2" \
  --duration 60 \
  --channels "0" \
  --rx_rate 2e6 \
  --rx_subdev "A:0" \
  --tx_rate 2e6 \
  --tx_subdev "A:0"
```

This example will test two full-duplex streams at 2 MS/s, for 60 seconds:

```
  /usr/local/lib/uhd/examples/benchmark_rate  \
  --args "addr=192.168.10.2" \
  --duration 60 \
  --channels "0,1" \
  --rx_rate 2e6 \
  --rx_subdev "A:0 A:1" \
  --tx_rate 2e6 \
  --tx_subdev "A:0 A:1"
```

This example will test two full-duplex streams at 12.5 MS/s, for 60 seconds:

```
  /usr/local/lib/uhd/examples/benchmark_rate  \
  --args "addr=192.168.10.2,master_clock_rate=25e6" \
  --duration 60 \
  --channels "0,1" \
  --rx_rate 12.5e6 \
  --rx_subdev "A:0 A:1" \
  --tx_rate 12.5e6 \
```

```
--tx_subdev "A:0 A:1"
```

When streaming samples over a 1 Gb transport link, the maximum accumulative rate for all channels is 25 MS/s with a `sc16` OTW format. To achieve higher streaming rates, it is recommended to use the 10 Gb interfaces.

NOTE: These examples require the `XG` FPGA image to be loaded.

This example will test one full-duplex stream using "RFA/A:0", at a rate of 61.44 MS/s, for 60 seconds:

```
/usr/local/lib/uhd/examples/benchmark_rate  \
--args "addr=192.168.10.2,master_clock_rate=61.44e6" \
--duration 60 \
--channels "0" \
--rx_rate 61.44e6 \
--rx_subdev "A:0" \
--tx_rate 61.44e6 \
--tx_subdev "A:0"
```

This example will test two full-duplex stream, at a rate of 30.72 MS/s, for 60 seconds:

```
/usr/local/lib/uhd/examples/benchmark_rate  \
--args "addr=192.168.10.2,master_clock_rate=61.44e6" \
--duration 60 \
--channels "0,1" \
--rx_rate 30.72e6 \
--rx_subdev "A:0 A:1" \
--tx_rate 30.72e6 \
--tx_subdev "A:0 A:1"
```

To avoid damaging the file system and causing any corruption, do not turn the device off with the power button without first shutting down the system. Use this command to cleanly and properly shut the system down:

```
shutdown --h now
```

The USRP E320 can be configured to power on and boot automatically when power is applied. By default, autoboot is disabled on all USRPs that support it. To control autoboot on the USRP E320, first determine the current value for `MCU_FLAGS[0]` by running `eeprom-dump`; the meaning of the `MCU_FLAGS[0]` is found in the UHD manual. The least significant bit when `MCU_FLAGS[0]` is viewed as a binary value controls the autoboot.

For example

```
root@ni-e320-XXXXXXX:~# eeprom-dump
-- PID/REV: e320 0002
-- MCU_FLAGS[0]: 00000008
-- MCU_FLAGS[1]: 00000000
-- MCU_FLAGS[2]: 00000000
-- MCU_FLAGS[3]: 00000000
-- Serial: XXXXXXX
-- eth_addr0: XX:XX:XX:XX:XX:XX
-- eth_addr1: XX:XX:XX:XX:XX:XX
-- eth_addr2: XX:XX:XX:XX:XX:XX
-- DT-Compat/MCU-Compat: 0000 0002
-- CRC: cbd79a61 (matches)
```

shows `-- MCU_FLAGS[0]: 00000008`; `0x08` (`0b00001000` in binary) indicates that autoboot is disabled. If this value were `0x09` (`0b00001001` in binary) it would indicate that autoboot is enabled because least significant bit is 1; same would be true if this value is `0x01` (`0b00000001` in binary).

To enable or disable autoboot, copy the existing value of `MCU_FLAGS[0]` retrieved by `eeprom-dump` into `<MCU_FLAGS[0]>` below and run the command:

- Disable autoboot on USRP E320 (sets least significant bit to 0), regardless of whether currently enabled or disabled:

```
root@ni-e320-XXXXXXX:~# eeprom-set-flags $((0x<MCU_FLAGS[0]> & ~0x1))
```

Thus, for the value noted above (autoboot is already disabled, so this command doesn't actually change anything):

```
root@ni-e320-XXXXXXX:~# eeprom-set-flags $((0x00000008 & ~0x1))
-- PID/REV: e320 0002
-- MCU_FLAGS[0]: 00000008
-- MCU_FLAGS[1]: 00000000
-- MCU_FLAGS[2]: 00000000
-- MCU_FLAGS[3]: 00000000
-- Serial: XXXXXXX
-- eth_addr0: XX:XX:XX:XX:XX:XX
-- eth_addr1: XX:XX:XX:XX:XX:XX
-- eth_addr2: XX:XX:XX:XX:XX:XX
-- DT-Compat/MCU-Compat: 0000 0002
-- CRC: cbd79a61 (matches)
-- Reading back
-- PID/REV: e320 0002
-- MCU_FLAGS[0]: 00000008
-- MCU_FLAGS[1]: 00000000
-- MCU_FLAGS[2]: 00000000
-- MCU_FLAGS[3]: 00000000
-- Serial: XXXXXXX
-- eth_addr0: XX:XX:XX:XX:XX:XX
-- eth_addr1: XX:XX:XX:XX:XX:XX
-- eth_addr2: XX:XX:XX:XX:XX:XX
-- DT-Compat/MCU-Compat: 0000 0002
-- CRC: 448fb572 (matches)
```

- Enable autoboot on USRP E320 (sets least significant bit to 1), regardless of whether currently enabled or disabled. For example when changing from autoboot disabled to enabled:

```
root@ni-e320-XXXXXXX:~# eeprom-set-flags $((0x<MCU_FLAGS[0]> | 0x1))
```

Thus, for the value noted above:

```
root@ni-e320-XXXXXXX:~# eeprom-set-flags $((0x00000008 | 0x1))
```

```
-- PID/REV: e320 0002
-- MCU_FLAGS[0]: 00000008
-- MCU_FLAGS[1]: 00000000
-- MCU_FLAGS[2]: 00000000
-- MCU_FLAGS[3]: 00000000
-- Serial: XXXXXXX
-- eth_addr0: XX:XX:XX:XX:XX:XX
-- eth_addr1: XX:XX:XX:XX:XX:XX
-- eth_addr2: XX:XX:XX:XX:XX:XX
-- DT-Compat/MCU-Compat: 0000 0002
-- CRC: cbd79a61 (matches)
-- Reading back
-- PID/REV: e320 0002
-- MCU_FLAGS[0]: 00000009
-- MCU_FLAGS[1]: 00000000
-- MCU_FLAGS[2]: 00000000
-- MCU_FLAGS[3]: 00000000
-- Serial: XXXXXXX
-- eth_addr0: XX:XX:XX:XX:XX:XX
-- eth_addr1: XX:XX:XX:XX:XX:XX
-- eth_addr2: XX:XX:XX:XX:XX:XX
-- DT-Compat/MCU-Compat: 0000 0002
-- CRC: 448fb572 (matches)
```

If setting this flag *does not* allow autoboot control on the USRP E320, then the device boot firmware needs to be updated. This update is accomplished via the following instructions.

On the USRP E320 via ssh or serial terminal, download the update MCU firmware and extract it:

```
root@ni-e320-XXXXXXX:~# curl https://files.ettus.com/binaries/misc/upgrade_mcu_neon_v1.1.7358-a190641-musl-glibc-rev3-7.tar.gz | tar zxf -
```

This will create a directory `upgrade_mcu_neon_v1.1.7358-a190641-musl-glibc-rev3-7`. Go into this directory and run the firmware flash script:

```
root@ni-e320-XXXXXXX:~# cd upgrade_mcu_neon_v1.1.7358-a190641-musl-glibc-rev3-7
root@ni-e320-XXXXXXX:~/upgrade_mcu_neon_v1.1.7358-a190641-musl-glibc-rev3-7# ./flash-firmware.sh
This script updates the microcontroller firmware (RO part). The change is
persistent across power cycles. Incorrect updates can only fixed be a manual
process which requires opening the enclosure.

Updating the microcontroller firmware (RO part) is only required if the Ettus
Research support told you to do so.

Press "y" to continue
```

At the prompt, press the `y` key to continue. Pressing any other key aborts the procedure:

```
Press "y" to continue n

aborting
root@ni-e320-317F9BF:~/upgrade_mcu_neon_v1.1.7358-a190641-musl-glibc-rev3-7#
```

Pressing the `y` key:

```
Press "y" to continue y

This script will flash ec-neon-rev3.RO.flat to the device
old RO version:    neon_vX.X.XXXX-XXXXXXX
new RO version:    neon_v1.1.7358-a190641

Press "y" to continue
```

At the prompt, press the `y` key *again* to continue. Pressing any other key aborts the procedure as before.

```
Press "y" to continue y

./ectool --interface=dev reboot_ec RW
./ectool --interface=dev flashread 0x0 65536 ec-neon-rev3.RO.flat.old
Reading 65536 bytes at offset 0...
done.
./ectool --interface=dev flasherase 0x0 65536
Erasing 65536 bytes at offset 0...
done.
./ectool --interface=dev flashwrite 0x0 ec-neon-rev3.RO.flat
Reading 49592 bytes from ec-neon-rev3.RO.flat...
Writing to offset 0...
Write size 112...
done.

copying new firmware files
'ec-neon-rev3.bin' -> '/lib/firmware/ni/ec-neon-rev3.bin'
'ec-neon-rev3.RW.bin' -> '/lib/firmware/ni/ec-neon-rev3.RW.bin'
root@ni-e320-317F9BF:~/upgrade_mcu_neon_v1.1.7358-a190641-musl-glibc-rev3-7#
```

Once the script is done, reboot the USRP (e.g., `shutdown -r now`), and when it comes up the autoboot flag should now work as desired. If these instructions *do not* work, then email support@ettus.com and ask for alternative instructions on how to update the USRP E320 RO and RW boot firmware such that this EEPROM flag setting is honored.

The default user is `root` and the password is empty (no password).

It is recommended to update the `root` password, which can be done with the command `passwd`:

Example Output:

```
root@ni-e320-serial:~# passwd
Changing password for root
New password:
Re-enter new password:
passwd: password changed.
```

In some streaming modes, the Intel I219-LM NIC can produce flow control and sequence errors. It is recommended to use a USB3 to 1 Gb Ethernet Adapter for hosts which have an I219-LM NIC.

Technical support for USRP hardware is available through email only. If the product arrived in a non−functional state or you require technical assistance, please contact support@ettus.com. Please allow 24 to 48 hours for response by email, depending on holidays and weekends, although we are often able to reply more quickly than that.

We also recommend that you subscribe to the community mailing lists. The mailing lists have a responsive and knowledgeable community of hundreds of developers and technical users who are located around the world. When you join the community, you will be connected to this group of people who can help you learn about SDR and respond to your technical and specific questions. Often your question can be answered quickly on the mailing lists. Each mailing list also provides an archive of all past conversations and discussions going back many years. Your question or problem may have already been addressed before, and a relevant or helpful solution may already exist in the archive.

Discussions involving the USRP hardware and the UHD software itself are best addressed through the **u?srp−-users** ?mailing list at http://usrp-users.ettus.com.

Discussions involving the use of GNU Radio with USRP hardware and UHD software are best addressed through the **d?iscuss−-gnuradio**? mailing list at https://lists.gnu.org/mailman/listinfo/discuss−gnuradio?.

Discussions involving the use of OpenBTS® with USRP hardware and UHD software are best addressed through the **o?penbts−-discuss?** mailing list at https://lists.sourceforge.net/lists/listinfo/openbts−discuss?.?

The support page on our website is located at https://www.ettus.com/support?. The Knowledge Base is located at ?https://kb.ettus.com?.

Every country has laws governing the transmission and reception of radio signals. Users are solely responsible for insuring they use their USRP system in compliance with all applicable laws and regulations. Before attempting to transmit and/or receive on any frequency, we recommend that you determine what licenses may be required and what restrictions may apply.

- NOTE: This USRP product is a piece of test equipment.

If you have any non−-technical questions related to your order, then please contact us by email at orders@ettus.com?, or by phone at +1−408−610−6399 (Monday-Friday, 8 AM - 5 PM, Pacific Time). Please be sure to include your order number and the serial number of your USRP.

Terms and conditions of sale can be accessed online at the following link: http://www.ettus.com/legal/terms-and-conditions-of-sale